

---

# ACM TEMPLATE

---

Orz

Last build at October 6, 2012

# Contents

<b>1</b>	<b>To Do List</b>	<b>3</b>
<b>2</b>	<b>注意事项</b>	<b>4</b>
<b>3</b>	<b>字符串处理</b>	<b>5</b>
3.1	*AC自动机	5
3.1.1	指针	5
3.1.2	非指针	6
3.2	后缀数组	7
3.2.1	DC3	7
3.2.2	DA	8
3.3	后缀三兄弟	8
3.3.1	例题	11
3.4	KMP	16
3.5	e-KMP	17
3.6	*Manacher	17
3.7	*字符串最小表示法	18
<b>4</b>	<b>数学</b>	<b>19</b>
4.1	模线性方程组	19
4.2	扩展GCD	19
4.3	矩阵	19
4.4	康拓展开	20
4.5	FFT	21
4.6	爬山法计算器	23
4.7	线性筛	25
<b>5</b>	<b>数据结构</b>	<b>26</b>
5.1	*Splay	26
5.2	*动态树	32
5.2.1	维护点权	32
5.3	可持久化线段树	35
5.4	treap正式版	38
5.5	树链剖分	40
5.5.1	点权	40
5.5.2	边权	45
<b>6</b>	<b>图论</b>	<b>49</b>
6.1	SAP四版	49
6.2	费用流三版	50
6.3	一般图匹配带花树	52
6.4	*二维平面图的最大流	54
<b>7</b>	<b>计算几何</b>	<b>58</b>
7.1	基本函数	58
7.1.1	Point定义	58
7.1.2	Line定义	58
7.1.3	距离: 两点距离	58
7.1.4	距离: 点到线段距离	58
7.1.5	面积: 多边形	59
7.1.6	判断: 线段相交	59
7.1.7	求解: 点到线最近点	59
7.2	重心	59
7.3	KD树	60
7.3.1	例题	61
7.4	半平面交	63
7.5	凸包	63
7.6	直线与凸包求交点	64
7.7	三维凸包	65
7.8	旋转卡壳	69
7.8.1	单个凸包	69
7.8.2	两个凸包	69

7.8.3	外接矩形	70
8	搜索	71
8.1	Dancing Links	71
9	杂物	74
9.1	高精度数	74
9.2	整数外挂	74
9.3	Java	75
9.3.1	优先队列	75
9.3.2	Map	75
9.3.3	sort	75

# 1 To Do List

测试DC3模板。。

所有带\*的内容。。。

可以从原来的模板里面继承一些好东西过来。

set,map,multiset等的搞基用法，以及注意事项。

## 2 注意事项

$10^6$ 数量级慎用后缀数组

TLE的时候要冷静哟。。

7k+的图计数 (Wc2012的communication)

思考的时候结合具体步骤来的话 会体会到一些不同的东西

C++与G++是很不一样的。。。

map套字符串是很慢的。。。

栈会被记录内存。。。

浮点数最短路要注意取 $\leq$ 来判断更新。。。

注意 long long

不要相信.size()

重复利用数组时 小心数组范围

先构思代码框架 每当实际拍马框架变化时 停手 重新思考

有时候四边形不等式也是帮得上忙的 dp 优化是可以水的

结构体里面带数组会非常慢,有时候 BFS 把数组压成数字会快很多。

```
1 | void fun(int a[])
2 | {
3 |     printf("%d\n", sizeof(a));
4 | }
```

结果是 sizeof(a[0]),如果传数组指针然后要清空的话不要用 sizeof。

sqrt 某些时候会出现 sqrt(-0.00)的问题。

## 3 字符串处理

### 3.1 \*AC自动机

#### 3.1.1 指针

```
1  const int CHAR=26;
2  const int TOTLEN=500000;
3  const int MAXLEN=1000000;
4  struct Vertex
5  {
6      Vertex *fail,*next[CHAR];
7      Vertex(){}
8      Vertex(bool flag)//为什么要这样写?
9      {
10         fail=0;
11         memset(next,0,sizeof(next));
12     }
13 };
14 int size;
15 Vertex vertex[TOTLEN+1];
16 void init()
17 {
18     vertex[0]=Vertex(0);
19     size=1;
20 }
21 void add(Vertex *pos,int cha)
22 {
23     vertex[size]=Vertex(0);
24     pos->next[cha]=&vertex[size++];
25 }
26 void add(vector<int> s)
27 {
28     int l=s.size();
29     Vertex *pos=&vertex[0];
30     for (int i=0; i<l; i++)
31     {
32         if (pos->next[s[i]]==NULL)
33             add(pos,s[i]);
34         pos=pos->next[s[i]];
35     }
36 }
37 void bfs()
38 {
39     queue<Vertex *> que;
40     Vertex *u=&vertex[0];
41     for (int i=0; i<CHAR; i++)
42         if (u->next[i]!=NULL)
43         {
44             que.push(u->next[i]);
45             u->next[i]->fail=u;
46         }
47         else
48             u->next[i]=u;
49     u->fail=NULL;
50     while (!que.empty())
51     {
52         u=que.front();
53         que.pop();
54         for (int i=0; i<CHAR; i++)
55             if (u->next[i]!=NULL)
56             {
57                 que.push(u->next[i]);
```

```

58         u->next[i]->fail=u->fail->next[i];
59     }
60     else
61         u->next[i]=u->fail->next[i];
62 }
63 }

```

### 3.1.2 非指针

```

1 struct Trie
2 {
3     int next[50][10],fail[50];
4     bool end[50];
5     int L,root;
6
7     int newNode()
8     {
9         for (int i = 0;i < 10;i++)
10             next[L][i] = -1;
11         end[L] = false;
12         return L++;
13     }
14
15     void Init()
16     {
17         L = 0;
18         root = newNode();
19     }
20
21     void Insert(char s[])
22     {
23         int now = root;
24         for (int i = 0;s[i] != 0;i++)
25         {
26             if (next[now][s[i]-'0'] == -1)
27                 next[now][s[i]-'0'] = newNode();
28             now = next[now][s[i]-'0'];
29         }
30         end[now] = true;
31     }
32
33     void Build()
34     {
35         queue<int> Q;
36         for (int i = 0;i < 10;i++)
37             if (next[root][i] == -1)
38                 next[root][i] = root;
39             else
40             {
41                 fail[next[root][i]] = root;
42                 Q.push(next[root][i]);
43             }
44         while (!Q.empty())
45         {
46             int now = Q.front();
47             Q.pop();
48             end[now] |= end[fail[now]];
49             for (int i = 0;i < 10;i++)
50                 if (next[now][i] == -1)
51                     next[now][i] = next[fail[now]][i];
52                 else
53                 {
54                     fail[next[now][i]] = next[fail[now]][i];
55                     Q.push(next[now][i]);

```

```

56         }
57     }
58 }
59 };

```

## 3.2 后缀数组

### 3.2.1 DC3

所有下标都是0 n-1, height[0]无意义。

```

1 //所有相关数组都要开三倍
2 const int maxn = 300010;
3 # define F(x) ((x)/3+((x)%3==1?0:tb))
4 # define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
5 int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
6 int c0(int *r, int a, int b)
7 {
8     return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
9 }
10 int c12(int k, int *r, int a, int b)
11 {
12     if (k == 2) return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
13     else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
14 }
15 void sort(int *r, int *a, int *b, int n, int m)
16 {
17     int i;
18     for (i = 0; i < n; i++) wv[i] = r[a[i]];
19     for (i = 0; i < m; i++) ws[i] = 0;
20     for (i = 0; i < n; i++) ws[wv[i]]++;
21     for (i = 1; i < m; i++) ws[i] += ws[i - 1];
22     for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
23     return;
24 }
25 void dc3(int *r, int *sa, int n, int m)
26 {
27     int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0,
        p;
28     r[n] = r[n + 1] = 0;
29     for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
30     sort(r + 2, wa, wb, tbc, m);
31     sort(r + 1, wb, wa, tbc, m);
32     sort(r, wa, wb, tbc, m);
33     for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
34         rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
35     if (p < tbc) dc3(rn, san, tbc, p);
36     else for (i = 0; i < tbc; i++) san[rn[i]] = i;
37     for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
38     if (n % 3 == 1) wb[ta++] = n - 1;
39     sort(r, wb, wa, ta, m);
40     for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
41     for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
42         sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
43     for (; i < ta; p++) sa[p] = wa[i++];
44     for (; j < tbc; p++) sa[p] = wb[j++];
45 }
46 //str和sa也要三倍
47 void da(int str[], int sa[], int rank[], int height[], int n, int m)
48 {
49     for (int i = n; i < n * 3; i++)
50         str[i] = 0;
51     dc3 (str , sa , n + 1 , m);
52     int i, j, k;

```



```

53     for (i = 0; i < n; i++)
54     {
55         sa[i] = sa[i + 1];
56         rank[sa[i]] = i;
57     }
58     for (i = 0, j = 0, k = 0; i < n; height[rank[i++]] = k)
59         if (rank[i] > 0)
60             for (k ? k-- : 0, j = sa[rank[i] - 1]; i + k < n && j + k < n
61                 &&
62                     str[i + k] == str[j + k]; k++) ;

```

### 3.2.2 DA

这份似乎就没啥要注意的了。

```

1  const int maxn = 200010;
2  int wx[maxn], wy[maxn], *x, *y, wss[maxn], wv[maxn];
3
4  bool cmp(int *r, int n, int a, int b, int l)
5  {
6      return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
7  }
8  void da(int str[], int sa[], int rank[], int height[], int n, int m)
9  {
10     int *s = str;
11     int *x=wx, *y=wy, *t, p;
12     int i, j;
13     for(i=0; i<m; i++) wss[i]=0;
14     for(i=0; i<n; i++) wss[x[i]=s[i]]++;
15     for(i=1; i<m; i++) wss[i]+=wss[i-1];
16     for(i=n-1; i>=0; i--) sa[--wss[x[i]]]=i;
17     for(j=1, p=1; p<n && j<n; j*=2, m=p)
18     {
19         for(i=n-j, p=0; i<n; i++) y[p++]=i;
20         for(i=0; i<n; i++) if(sa[i]-j>=0) y[p++]=sa[i]-j;
21         for(i=0; i<n; i++) wv[i]=x[y[i]];
22         for(i=0; i<m; i++) wss[i]=0;
23         for(i=0; i<n; i++) wss[wv[i]]++;
24         for(i=1; i<m; i++) wss[i]+=wss[i-1];
25         for(i=n-1; i>=0; i--) sa[--wss[wv[i]]]=y[i];
26         for(t=x, x=y, y=t, p=1, i=1, x[sa[0]]=0; i<n; i++)
27             x[sa[i]]=cmp(y, n, sa[i-1], sa[i], j)?p-1:p++;
28     }
29     for(int i=0; i<n; i++) rank[sa[i]]=i;
30     for(int i=0, j=0, k=0; i<n; height[rank[i++]]=k)
31         if(rank[i]>0)
32             for(k?k--:0, j=sa[rank[i]-1]; i+k < n && j+k < n && str[i+k]==str
33                 [j+k]; k++);

```

## 3.3 后缀三兄弟

```

1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  using namespace std;
5  const int CHAR = 26;
6  const int MAXN = 100000;
7  struct SAM_Node
8  {
9      SAM_Node *fa, *next[CHAR];
10     int len;

```

```

11     int id,pos;
12     SAM_Node() {}
13     SAM_Node(int _len)
14     {
15         fa = 0;
16         len = _len;
17         memset(next,0,sizeof(next));
18     }
19 };
20 SAM_Node SAM_node[MAXN * 2], *SAM_root, *SAM_last;
21 int SAM_size;
22 SAM_Node *newSAM_Node(int len)
23 {
24     SAM_node[SAM_size] = SAM_Node(len);
25     SAM_node[SAM_size].id=SAM_size;
26     return &SAM_node[SAM_size++];
27 }
28 SAM_Node *newSAM_Node(SAM_Node *p)
29 {
30     SAM_node[SAM_size] = *p;
31     SAM_node[SAM_size].id=SAM_size;
32     return &SAM_node[SAM_size++];
33 }
34 void SAM_init()
35 {
36     SAM_size = 0;
37     SAM_root = SAM_last = newSAM_Node(0);
38     SAM_node[0].pos=0;
39 }
40 void SAM_add(int x,int len)
41 {
42     SAM_Node *p = SAM_last, *np = newSAM_Node(p->len + 1);
43     np->pos=len;
44     SAM_last = np;
45     for (; p && !p->next[x]; p = p->fa)
46         p->next[x] = np;
47     if (!p)
48     {
49         np->fa = SAM_root;
50         return ;
51     }
52     SAM_Node *q = p->next[x];
53     if (q->len == p->len + 1)
54     {
55         np->fa = q;
56         return ;
57     }
58     SAM_Node *nq = newSAM_Node(q);
59     nq->len = p->len + 1;
60     q->fa = nq;
61     np->fa = nq;
62     for (; p && p->next[x] == q; p = p->fa)
63         p->next[x] = nq;
64 }
65 void SAM_build(char *s)
66 {
67     SAM_init();
68     int l = strlen(s);
69     for (int i = 0; i < l; i++)
70         SAM_add(s[i] - 'a',i+1);
71 }
72
73 SAM_Node * SAM_add(SAM_Node *p, int x, int len)

```

```

74 {
75     SAM_Node *np = newSAM_Node(p->len + 1);
76     np->pos = len;
77     SAM_last = np;
78     for (; p && !p->next[x]; p = p->fa)
79         p->next[x] = np;
80     if (!p)
81     {
82         np->fa = SAM_root;
83         return np;
84     }
85     SAM_Node *q = p->next[x];
86     if (q->len == p->len + 1)
87     {
88         np->fa = q;
89         return np;
90     }
91     SAM_Node *nq = newSAM_Node(q);
92     nq->len = p->len + 1;
93     q->fa = nq;
94     np->fa = nq;
95     for (; p && p->next[x] == q; p = p->fa)
96         p->next[x] = nq;
97     return np;
98 }
99 void SAM_build(char *s)//多串建立 注意SAM_init()的调用
100 {
101     int l = strlen(s);
102     SAM_Node *p = SAM_root;
103     for (int i = 0; i < l; i++)
104     {
105         if (!p->next[s[i] - 'a'] || !(p->next[s[i] - 'a']->len == i + 1))
106             p = SAM_add(p, s[i] - 'a', i + 1);
107         else
108             p = p->next[s[i] - 'a'];
109     }
110 }
111
112 struct ST_Node
113 {
114     ST_Node *next[CHAR], *fa;
115     int len, pos;
116 } ST_node[MAXN*2], *ST_root;
117 int Sufpos[MAXN];
118 void ST_add(int u, int v, int chr, int len)
119 {
120     ST_node[u].next[chr] = &ST_node[v];
121     ST_node[v].len = len;
122 }
123 void init(int n)
124 {
125     for (int i = 0; i < n; i++)
126     {
127         ST_node[i].pos = -1;
128         ST_node[i].fa = 0;
129         memset(ST_node[i].next, 0, sizeof(ST_node[i].next));
130     }
131     ST_node[0].pos = 0;
132     ST_root = &ST_node[0];
133 }
134 void ST_build(char *s)
135 {
136     int n = strlen(s);

```

```

137     reverse(s,s+n);
138     SAM_build(s);
139     init(SAM_size);
140     for (int i=1;i<SAM_size;i++)
141     {
142         ST_add(SAM_node[i].fa->id,SAM_node[i].id,s[SAM_node[i].pos-SAM_node[
            i].fa->len-1]-'a',SAM_node[i].len-SAM_node[i].fa->len);
143         if (SAM_node[i].pos==SAM_node[i].len)
144         {
145             Sufpos[n-SAM_node[i].pos+1]=i;
146             ST_node[i].pos=n-SAM_node[i].pos+1;
147         }
148     }
149 }
150
151 int rank[MAXN],sa[MAXN+1];
152 int height[MAXN];
153 int L;
154 void ST_dfs(ST_Node *p)
155 {
156     if (p->pos!=-1)
157         sa[L++]=p->pos;
158     for (int i=0;i<CHAR;i++)
159         if (p->next[i])
160             ST_dfs(p->next[i]);
161 }
162 char s[MAXN+1];
163 int main()
164 {
165     gets(s);
166     ST_build(s);
167     L=0;
168     ST_dfs(ST_root);
169     int n=strlen(s);
170     for (int i=0; i<n; i++)
171         sa[i]=sa[i+1]-1;
172     for (int i=0; i<n; i++)
173         rank[sa[i]]=i;
174     reverse(s,s+n);
175     for (int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
176         if (rank[i])
177             for (k?k--:0,j=sa[rank[i]-1]; s[i+k]==s[j+k]; k++);
178 }

```

### 3.3.1 例题

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {

```

```

19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
42     SAM_root = SAM_last = newSAM_Node(0);
43 }
44 void SAM_add(int x,int len)
45 {
46     SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47     SAM_last = np;
48     for (; p&&!p->next[x]; p=p->fa)
49         p->next[x] = np;
50     if (!p)
51     {
52         np->fa = SAM_root;
53         return;
54     }
55     SAM_Node *q = p->next[x];
56     if (q->len == p->len+1)
57     {
58         np->fa = q;
59         return;
60     }
61     SAM_Node *nq = newSAM_Node(q);
62     nq->len = p->len+1;
63     q->fa = nq;
64     np->fa = nq;
65     for (; p&&p->next[x] == q; p = p->fa)
66         p->next[x] = nq;
67 }
68 int getid(char ch)
69 {
70     return ch-'a';
71 }
72 void SAM_build(char *s)
73 {
74     SAM_init();
75     int l = strlen(s);
76     for (int i = 0; i < l; i++)
77         SAM_add(getid(s[i]),i+1);
78 }
79 char s[10][MAXN+1];
80 int ans;
81 int head[MAXN*2];

```

```

82 struct Edge
83 {
84     int to,next;
85 } edge[MAXN*2];
86 int M;
87 int n;
88 void add_edge(int u,int v)
89 {
90     edge[M].to=v;
91     edge[M].next=head[u];
92     head[u]=M++;
93 }
94 void dfs(int u)
95 {
96     for (int i=head[u]; i!=-1; i=edge[i].next)
97     {
98         int v=edge[i].to;
99         dfs(v);
100         for (int j=0; j<n-1; j++)
101             SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);
102     }
103     int tmp=SAM_node[u].len;
104     for (int i=0; i<n-1; i++)
105         tmp=min(tmp,SAM_node[u].mat[i]);
106     ans=max(ans,tmp);
107 }
108 int main()
109 {
110
111     while (scanf("%s",s[n])!=EOF)
112         n++;
113     int L=strlen(s[0]);
114     ans=M=0;
115     SAM_build(s[0]);
116     for (int j=1; j<n; j++)
117     {
118         int l=strlen(s[j]),len=0;
119         SAM_Node *p=SAM_root;
120         for (int i=0; i<l; i++)
121         {
122             if (p->next[getid(s[j][i])])
123             {
124                 p=p->next[getid(s[j][i])];
125                 p->mat[j-1]=max(p->mat[j-1],++len);
126             }
127             else
128             {
129                 while (p && !p->next[getid(s[j][i])])
130                     p=p->fa;
131                 if (!p)
132                 {
133                     p=SAM_root;
134                     len=0;
135                 }
136                 else
137                 {
138                     len=p->len+1;
139                     p=p->next[getid(s[j][i])];
140                 }
141                 p->mat[j-1]=max(p->mat[j-1],len);
142             }
143         }
144     }

```

```

145     memset(head,-1,4*SAM_size);
146     for (int i=1; i<SAM_size; i++)
147         add_edge(SAM_node[i].fa->id,i);
148     dfs(0);
149     printf("%d\n",ans);
150     return 0;
151 }

```

## LCS2

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {
19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
42     SAM_root = SAM_last = newSAM_Node(0);
43 }
44 void SAM_add(int x,int len)
45 {
46     SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47     SAM_last = np;
48     for (; p&&!p->next[x]; p=p->fa)
49         p->next[x] = np;
50     if (!p)
51     {
52         np->fa = SAM_root;
53         return;

```

```

54     }
55     SAM_Node *q = p->next[x];
56     if (q->len == p->len+1)
57     {
58         np->fa = q;
59         return;
60     }
61     SAM_Node *nq = newSAM_Node(q);
62     nq->len = p->len+1;
63     q->fa = nq;
64     np->fa = nq;
65     for (; p&& p->next[x] == q; p = p->fa)
66         p->next[x] = nq;
67 }
68 int getid(char ch)
69 {
70     return ch-'a';
71 }
72 void SAM_build(char *s)
73 {
74     SAM_init();
75     int l = strlen(s);
76     for (int i = 0; i < l; i++)
77         SAM_add(getid(s[i]),i+1);
78 }
79 char s[MAXN+1];
80 int ans;
81 int head[MAXN*2];
82 struct Edge
83 {
84     int to,next;
85 } edge[MAXN*2];
86 int M;
87 int n;
88 void add_edge(int u,int v)
89 {
90     edge[M].to=v;
91     edge[M].next=head[u];
92     head[u]=M++;
93 }
94 void dfs(int u)
95 {
96     for (int i=head[u]; i!=-1; i=edge[i].next)
97     {
98         int v=edge[i].to;
99         /*for (int j=0; j<n; j++)
100             SAM_node[v].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);*/
101         dfs(v);
102         for (int j=0; j<n; j++)
103             SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);
104     }
105     int tmp=SAM_node[u].len;
106     for (int i=0; i<n; i++)
107         tmp=min(tmp,SAM_node[u].mat[i]);
108     ans=max(ans,tmp);
109 }
110 int main()
111 {
112     //freopen("in.txt","r",stdin);
113     //freopen("out.txt","w",stdout);
114     n=0;
115     gets(s);
116     SAM_build(s);

```



```

117 while (gets(s))
118 {
119     int l=strlen(s),len=0;
120     SAM_Node *p=SAM_root;
121     for (int i=0; i<l; i++)
122     {
123         if (p->next[getid(s[i])])
124         {
125             p=p->next[getid(s[i])];
126             p->mat[n]=max(p->mat[n],++len);
127         }
128         else
129         {
130             while (p && !p->next[getid(s[i])])
131                 p=p->fa;
132             if (!p)
133             {
134                 p=SAM_root;
135                 len=0;
136             }
137             else
138             {
139                 len=p->len+1;
140                 p=p->next[getid(s[i])];
141             }
142             p->mat[n]=max(p->mat[n],len);
143         }
144         //printf("%d %d %d\n",i,len,p->id);
145     }
146     n++;
147 }
148 memset(head,-1,4*SAM_size);
149 for (int i=1; i<SAM_size; i++)
150     add_edge(SAM_node[i].fa->id,i);
151 dfs(0);
152 printf("%d\n",ans);
153 return 0;
154 }

```

### 3.4 KMP

求A[0..i]的一个后缀最多能匹配B的前缀多长。先对B进行自匹配然后与A匹配。KMP[i]就是对应答案，p[i]+1是B[0..i]的一个后缀最多能匹配B的前缀多长。

```

1 //自匹配过程
2 int j;
3 p[0] = j = -1;
4 for (int i = 1; i < lb; i++)
5 {
6     while (j >= 0 && b[j + 1] != b[i]) j = p[j];
7     if (b[j + 1] == b[i]) j ++;
8     p[i] = j;
9 }
10 //下面是匹配过程
11 j = -1;
12 for (int i = 0; i < la; i++)
13 {
14     while (j >= 0 && b[j + 1] != a[i]) j = p[j];
15     if (b[j + 1] == a[i]) j ++;
16     KMP[i] = j + 1;
17 }

```

### 3.5 e-KMP

求A[i..len-1]和B的最长公共前缀有多长。先对B进行自匹配然后与A匹配。eKMP[i]就是对应答案。p[i]是B[i..len-1]和B的最长公共前缀有多长。

```
1 //自匹配过程
2 int j = 0;
3 while (j < lb && b[j] == b[j + 1])
4     j++;
5 p[0] = lb, p[1] = j;
6 int k = 1;
7 for (int i = 2; i < lb; i++)
8 {
9     int Len = k + p[k] - 1, L = p[i - k];
10    if (L < Len - i + 1)
11        p[i] = L;
12    else
13    {
14        j = max(0, Len - i + 1);
15        while (i + j < lb && b[i + j] == b[j])
16            j++;
17        p[i] = j, k = i;
18    }
19 }
20 //下面是匹配过程
21 j = 0;
22 while (j < la && j < lb && a[j] == b[j])
23     j++;
24 eKMP[0] = j;
25 k = 0;
26 for (int i = 1; i < la; i++)
27 {
28     int Len = k + eKMP[k] - 1, L = p[i - k];
29     if (L < Len - i + 1)
30         eKMP[i] = L;
31     else
32     {
33         j = max(0, Len - i + 1);
34         while (i + j < la && j < lb && a[i + j] == b[j])
35             j++;
36         eKMP[i] = j, k = i;
37     }
38 }
```

### 3.6 \*Manacher

待整理

```
1 char s[1000], a[3000];
2 int p[3000], len, l, pnow, pid, res, resid;
3
4 int main()
5 {
6     while (scanf("%s", s) != EOF)
7     {
8         len = strlen(s);
9         l = 0;
10        a[l++] = '.';
11        a[l++] = ',';
12        for (int i = 0; i < len; i++)
13        {
14            a[l++] = s[i];
15            a[l++] = ',';
16        }
17    }
```

```

17     pnow = 0;
18     res = 0;
19     for (int i = 1; i < l; i++)
20     {
21         if (pnow > i)
22             p[i] = min(p[2*pid-i], pnow-i);
23         else
24             p[i] = 1;
25         for (; a[i-p[i]] == a[i+p[i]]; p[i]++);
26         if (i+p[i] > pnow)
27         {
28             pnow = i+p[i];
29             pid = i;
30         }
31         if (p[i] > res)
32         {
33             res = p[i];
34             resid = i;
35         }
36     }
37     for (int i = resid-res+2; i < resid+res-1; i += 2)
38         printf("%c", a[i]);
39     printf("\n");
40 }
41 return 0;
42 }

```

### 3.7 \*字符串最小表示法

```

1 int Gao(char a[], int len)
2 {
3     int i = 0, j = 1, k = 0;
4     while (i < len && j < len && k < len)
5     {
6         int cmp = a[(j+k)%len] - a[(i+k)%len];
7         if (cmp == 0)
8             k++;
9         else
10        {
11            if (cmp > 0)
12                j += k+1;
13            else
14                i += k+1;
15            if (i == j) j++;
16            k = 0;
17        }
18    }
19    return min(i, j);
20 }

```

## 4 数学

### 4.1 模线性方程组

```
1 //有更新
2 int m[10],a[10]; //模数m 余数a
3 bool solve(int &m0,int &a0,int m,int a) //模线性方程组
4 {
5     int y,x;
6     int g=ex_gcd(m0,m,x,y);
7     if (abs(a-a0)%g) return 0;
8     x*=(a-a0)/g;
9     x%=m/g;
10    a0=(x*m0+a0);
11    m0*=m/g;
12    a0%=m0;
13    if (a0<0) a0+=m0;
14    return 1;
15 }
16 int MLES()
17 {
18     bool flag=1;
19     int m0=1,a0=0;
20     for (int i=0; i<n; i++)
21         if (!solve(m0,a0,m[i],a[i]))
22             {
23                 flag=0;
24                 break;
25             }
26     if (flag)
27         return a0;
28     else
29         return -1;
30 }
```

### 4.2 扩展GCD

求 $ax+by=\gcd(a,b)$ 的一组解

```
1 long long ex_gcd(long long a,long long b,long long &x,long long &y)
2 {
3     if (b)
4     {
5         long long ret = ex_gcd(b,a%b,x,y),tmp = x;
6         x = y;
7         y = tmp-(a/b)*y;
8         return ret;
9     }
10    else
11    {
12        x = 1;
13        y = 0;
14        return a;
15    }
16 }
```

### 4.3 矩阵

```
1 struct Matrix
2 {
3     int a[52][52];
4     Matrix operator * (const Matrix &b) const
5     {
```

```

6         Matrix res;
7         for (int i = 0; i < 52; i++)
8             for (int j = 0; j < 52; j++)
9                 {
10                     res.a[i][j] = 0;
11                     for (int k = 0; k < 52; k++)
12                         res.a[i][j] += a[i][k] * b.a[k][j];
13                 }
14         return res;
15     }
16     Matrix operator ^ (int y) const
17     {
18         Matrix res, x;
19         for (int i = 0; i < 52; i++)
20             {
21                 for (int j = 0; j < 52; j++)
22                     res.a[i][j] = 0, x.a[i][j] = a[i][j];
23                 res.a[i][i] = 1;
24             }
25         for (; y; y >>= 1, x = x * x)
26             if (y & 1)
27                 res = res * x;
28         return res;
29     }
30 };

```

#### 4.4 康拓展开

```

1  const int PermSize = 12;
2  int factory[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880,
3  3628800, 39916800};
4  int Cantor(int a[])
5  {
6      int i, j, counted;
7      int result = 0;
8      for (i = 0; i < PermSize; ++i)
9          {
10             counted = 0;
11             for (j = i + 1; j < PermSize; ++j)
12                 if (a[i] > a[j])
13                     ++counted;
14             result = result + counted * factory[PermSize - i - 1];
15         }
16     return result;
17 }
18 bool h[13];
19
20 void UnCantor(int x, int res[])
21 {
22     int i, j, l, t;
23     for (i = 1; i <= 12; i++)
24         h[i] = false;
25     for (i = 1; i <= 12; i++)
26     {
27         t = x / factory[12 - i];
28         x -= t * factory[12 - i];
29         for (j = 1, l = 0; l <= t; j++)
30             if (!h[j]) l++;
31         j--;
32         h[j] = true;
33         res[i - 1] = j;
34     }
35 }

```

## 4.5 FFT

```
1  const double PI= acos(-1.0);
2  struct vir
3  {
4      double re,im; //实部和虚部
5      vir(double a=0,double b=0)
6      {
7          re=a;
8          im=b;
9      }
10     vir operator +(const vir &b)
11     {return vir(re+b.re,im+b.im);}
12     vir operator -(const vir &b)
13     {return vir(re-b.re, im-b.im);}
14     vir operator *(const vir &b)
15     {return vir(re*b.re-im*b.im , re*b.im+im*b.re);}
16 };
17 vir x1[200005],x2[200005];
18 void change(vir *x,int len,int loglen)
19 {
20     int i,j,k,t;
21     for(i=0;i<len;i++)
22     {
23         t=i;
24         for(j=k=0; j<loglen; j++,t>>=1)
25             k= (k<<1)|(t&1);
26         if(k<i)
27         {
28             // printf("%d %d\n",k,i);
29             vir wt=x[k];
30             x[k]=x[i];
31             x[i]=wt;
32         }
33     }
34 }
35 void fft(vir *x,int len,int loglen)
36 {
37     int i,j,t,s,e;
38     change(x,len,loglen);
39     t=1;
40     for(i=0;i<loglen;i++,t<=<=1)
41     {
42         s=0;
43         e=s+t;
44         while(s<len)
45         {
46             vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
47             for(j=s;j<s+t;j++)
48             {
49                 a=x[j];
50                 b=x[j+t]*wn;
51                 x[j]=a+b;
52                 x[j+t]=a-b;
53                 wn=wn*wo;
54             }
55             s=e+t;
56             e=s+t;
57         }
58     }
59 }
60 void dit_fft(vir *x,int len,int loglen)
61 {
```

```

62     int i,j,s,e,t=1<<loglen;
63     for(i=0;i<loglen;i++)
64     {
65         t>>=1;
66         s=0;
67         e=s+t;
68         while(s<len)
69         {
70             vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
71             for(j=s;j<s+t;j++)
72             {
73                 a=x[j]+x[j+t];
74                 b=(x[j]-x[j+t])*wn;
75                 x[j]=a;
76                 x[j+t]=b;
77                 wn=wn*wo;
78             }
79             s=e+t;
80             e=s+t;
81         }
82     }
83     change(x,len,loglen);
84     for(i=0;i<len;i++)
85         x[i].re/=len;
86 }
87 int main()
88 {
89     char a[100005],b[100005];
90     int i,len1,len2,len,loglen;
91     int t,over;
92     while(scanf("%s%s",a,b)!=EOF)
93     {
94         len1=strlen(a)<<1;
95         len2=strlen(b)<<1;
96         len=1;loglen=0;
97         while(len<len1)
98         {
99             len<=1;    loglen++;
100         }
101         while(len<len2)
102         {
103             len<=1;    loglen++;
104         }
105         for(i=0;a[i];i++)
106         {
107             x1[i].re=a[i]-'0';
108             x1[i].im=0;
109         }
110         for(;i<len;i++)
111             x1[i].re=x1[i].im=0;
112         for(i=0;b[i];i++)
113         {
114             x2[i].re=b[i]-'0';
115             x2[i].im=0;
116         }
117         for(;i<len;i++)
118             x2[i].re=x2[i].im=0;
119         fft(x1,len,loglen);
120         fft(x2,len,loglen);
121         for(i=0;i<len;i++)
122             x1[i] = x1[i]*x2[i];
123         dit_fft(x1,len,loglen);
124         for(i=(len1+len2)/2-2,over=len=0;i>=0;i--)

```

```

125     {
126         t=(int)(x1[i].re+over+0.5);
127         a[len++]= t%10;
128         over = t/10;
129     }
130     while(over)
131     {
132         a[len++]=over%10;
133         over/=10;
134     }
135     for(len--;len>=0&&!a[len];len--);
136     if(len<0)
137         putchar('0');
138     else
139         for(;len>=0;len--)
140             putchar(a[len]+'0');
141     putchar('\n');
142 }
143 return 0;
144 }

```

## 4.6 爬山法计算器

注意灵活运用。

双目运算符在calc()中，左结合单目运算符在P()中，右结合单目运算符在calc\_exp中。（但是还没遇到过。。）

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <string>
6  using namespace std;
7
8  char s[100000];
9  int n,cur;
10 const string OP = "+-*/";
11
12 char next_char()
13 {
14     if (cur >= n) return EOF;
15     return s[cur];
16 }
17
18 int get_priority(char ch)
19 {
20     if (ch == '*' || ch == '/') return 2;
21     return 1;
22 }
23
24 int P();
25
26 int calc(int a,char op,int b)
27 {
28     if (op == '+')
29         return a+b;
30     if (op == '-')
31         return a-b;
32     if (op == '*')
33         return a*b;
34 }
35
36 int calc_exp(int p)
37 {

```



```

38     int a = P();
39     while ((OP.find(next_char()) != OP.npos) && (get_priority(next_char())
40         >= p))
41     {
42         char op = next_char();
43         cur++;
44         a = calc(a,op,calc_exp(get_priority(op)+1));
45     }
46     return a;
47 }
48 int totvar,m,var[26],varid[26];
49
50 int P()
51 {
52     if (next_char() == '-')
53     {
54         cur++;
55         return -P();
56     }
57     else if (next_char() == '+')
58     {
59         cur++;
60         return P();
61     }
62     else if (next_char() == '(')
63     {
64         cur++;
65         int res = calc_exp(0);
66         cur++;
67         return res;
68     }
69     else
70     {
71         cur++;
72         //cout << "getvar at " << cur << ' ' << var[varid[s[cur]-'a']] <<
73             endl;
74         return var[varid[s[cur-1]-'a']];
75     }
76 }
77 int id[26],minid;
78
79 int main()
80 {
81     while (true)
82     {
83         scanf("%d",&totvar,&var[0]);
84         if (totvar == 0 && var[0] == 0) break;
85         for (int i = 1;i < totvar;i++)
86             scanf("%d",&var[i]);
87         scanf("%d",&m);
88         scanf("%s",s);
89         for (int i = 0;i < 26;i++)
90             id[i] = -1;
91         minid = 0;
92         n = strlen(s);
93         for (int i = 0;i < n;i++)
94             if (s[i] >= 'a' && s[i] <= 'z')
95             {
96                 if (id[s[i]-'a'] == -1)
97                 {
98                     id[s[i]-'a'] = minid;

```

```

99         minid++;
100     }
101     s[i] = 'a'+id[s[i]-'a'];
102 }
103 for (int i = 0;i < totvar;i++)
104     varid[i] = i;
105 int res = 0;
106 do
107 {
108     cur = 0;
109     int tmp = calc_exp(0);
110     if (tmp == m)
111     {
112         res++;
113         break;
114     }
115 }
116 while (next_permutation(varid,varid+totvar));
117 //puts(s);
118 if (res > 0)
119     puts("YES");
120 else
121     puts("NO");
122 }
123 return 0;
124 }

```

## 4.7 线性筛

我弱逼。

```

1 void getprime()
2 {
3     tot = 0;
4     memset(isprime,true,sizeof(isprime));
5     for (int i = 2;i <= 40000000;i++)
6     {
7         if (isprime[i] == true)
8         {
9             tot++;
10            prime[tot] = i;
11        }
12        for (int j = 1;j <= tot && i*prime[j] <= 40000000;j++)
13        {
14            isprime[i*prime[j]] = false;
15            if (i%prime[j] == 0) break;
16        }
17    }
18 }

```

## 5 数据结构

### 5.1 \*Splay

持续学习中。

注意节点的size值不一定是真实的值！如果有需要需要特别维护！

1. 旋转和Splay操作
2. rank操作
3. insert操作（。。很多题目都有）
4. del操作（郁闷的出纳员）
5. 由数组建立Splay
6. 前驱后继（营业额统计）
7. Pushdown Pushup的位置
8. \*。。。暂时想不起了

节点定义。。

```
1 | const int MaxN = 50003;
2 |
3 | struct Node
4 | {
5 |     int size, key;
6 |
7 |     Node *c[2];
8 |     Node *p;
9 | } mem[MaxN], *cur, *nil;
```

无内存池的几个初始化函数。

```
1 | Node *newNode(int v, Node *p)
2 | {
3 |     cur->c[0] = cur->c[1] = nil, cur->p = p;
4 |     cur->size = 1;
5 |     cur->key = v;
6 |     return cur++;
7 | }
8 |
9 | void Init()
10 | {
11 |     cur = mem;
12 |     nil = newNode(0, cur);
13 |     nil->size = 0;
14 | }
```

带内存池的几个函数。

```
1 | int emp[MaxN], totemp;
2 |
3 | Node *newNode(int v, Node *p)
4 | {
5 |     cur = mem + emp[--totemp];
6 |     cur->c[0] = cur->c[1] = nil, cur->p = p;
7 |     cur->size = 1;
8 |     cur->key = v;
9 |     return cur;
10 | }
11 |
```

```

12 void Init()
13 {
14     for (int i = 0; i < MaxN; ++i)
15         emp[i] = i;
16     totemp = MaxN;
17     cur = mem + emp[--totemp];
18     nil = newNode(0, cur);
19     nil->size = 0;
20 }
21
22 void Recycle(Node *p)
23 {
24     if (p == nil) return;
25     Recycle(p->c[0]), Recycle(p->c[1]);
26     emp[totemp++] = p - mem;
27 }

```

基本的Splay框架。维护序列用。  
一切下标从0开始。

```

1 struct SplayTree
2 {
3     Node *root;
4     void Init()
5     {
6         root = nil;
7     }
8     void Pushup(Node *x)
9     {
10         if (x == nil) return;
11         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
12         x->size = x->c[0]->size + x->c[1]->size + 1;
13     }
14     void Pushdown(Node *x)
15     {
16         if (x == nil) return;
17         //do something
18     }
19     void Rotate(Node *x, int f)
20     {
21         if (x == nil) return;
22         Node *y = x->p;
23         y->c[f ^ 1] = x->c[f], x->p = y->p;
24         if (x->c[f] != nil)
25             x->c[f]->p = y;
26         if (y->p != nil)
27             y->p->c[y->p->c[1] == y] = x;
28         x->c[f] = y, y->p = x;
29         Pushup(y);
30     }
31     void Splay(Node *x, Node *f)
32     {
33         while (x->p != f)
34         {
35             Node *y = x->p;
36             if (y->p == f)
37                 Rotate(x, x == y->c[0]);
38             else
39             {
40                 int fd = y->p->c[0] == y;
41                 if (y->c[fd] == x)
42                     Rotate(x, fd ^ 1), Rotate(x, fd);
43                 else

```

```

44         Rotate(y, fd), Rotate(x, fd);
45     }
46 }
47 Pushup(x);
48 if (f == nil)
49     root = x;
50 }
51 void Select(int k, Node *f)
52 {
53     Node *x = root;
54     Pushdown(x);
55     int tmp;
56     while ((tmp = x->c[0]->size) != k)
57     {
58         if (k < tmp)    x = x->c[0];
59         else
60             x = x->c[1], k -= tmp + 1;
61         Pushdown(x);
62     }
63     Splay(x, f);
64 }
65 void Select(int l, int r)
66 {
67     Select(l, nil), Select(r + 2, root);
68 }
69 Node *Make_tree(int a[], int l, int r, Node *p)
70 {
71     if (l > r)    return nil;
72     int mid = l + r >> 1;
73     Node *x = newNode(a[mid], p);
74     x->c[0] = Make_tree(a, l, mid - 1, x);
75     x->c[1] = Make_tree(a, mid + 1, r, x);
76     Pushup(x);
77     return x;
78 }
79 void Insert(int pos, int a[], int n)
80 {
81     Select(pos, nil), Select(pos + 1, root);
82     root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
83     Splay(root->c[1]->c[0], nil);
84 }
85 void Insert(int v)
86 {
87     Node *x = root, *y = nil;
88     while (x != nil)
89     {
90         y = x;
91         y->size++;
92         x = x->c[v >= x->key];
93     }
94     y->c[v >= y->key] = x = newNode(v, y);
95     Splay(x, nil);
96 }
97 void Remove(int l, int r)
98 {
99     Select(l, r);
100     //Recycle(root->c[1]->c[0]);
101     root->c[1]->c[0] = nil;
102     Splay(root->c[1], nil);
103 }
104 };

```

例题：旋转区间赋值求和求最大子序列。

注意打上懒标记后立即Pushup。Pushup(root->c[1]->c[0]),Pushup(root->c[1]),Pushup(root);

```

1 void Pushup(Node *x)
2 {
3     if (x == nil) return;
4     Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
5     x->size = x->c[0]->size+x->c[1]->size+1;
6
7     x->sum = x->c[0]->sum+x->c[1]->sum+x->key;
8     x->lsum = max(x->c[0]->lsum,x->c[0]->sum+x->key+max(0,x->c[1]->lsum)
9         );
10    x->rsum = max(x->c[1]->rsum,x->c[1]->sum+x->key+max(0,x->c[0]->rsum)
11        );
12    x->maxsum = max(max(x->c[0]->maxsum,x->c[1]->maxsum),x->key+max(0,x
13        ->c[0]->rsum)+max(0,x->c[1]->lsum));
14 }
15 void Pushdown(Node *x)
16 {
17     if (x == nil) return;
18     if (x->rev)
19     {
20         x->rev = 0;
21         x->c[0]->rev ^= 1;
22         x->c[1]->rev ^= 1;
23         swap(x->c[0],x->c[1]);
24
25         swap(x->lsum,x->rsum);
26     }
27     if (x->same)
28     {
29         x->same = false;
30         x->key = x->lazy;
31         x->sum = x->key*x->size;
32         x->lsum = x->rsum = x->maxsum = max(x->key,x->sum);
33         x->c[0]->same = true, x->c[0]->lazy = x->key;
34         x->c[1]->same = true, x->c[1]->lazy = x->key;
35     }
36 }
37
38 int main()
39 {
40     int totcas;
41     scanf("%d",&totcas);
42     for (int cas = 1;cas <= totcas;cas++)
43     {
44         Init();
45         sp.Init();
46         nil->lsum = nil->rsum = nil->maxsum = -Inf;
47         sp.Insert(0);
48         sp.Insert(0);
49
50         int n,m;
51         scanf("%d%d",&n,&m);
52         for (int i = 0;i < n;i++)
53             scanf("%d",&a[i]);
54         sp.Insert(0,a,n);
55
56         for (int i = 0;i < m;i++)
57         {
58             int pos,tot,c;
59             scanf("%s",buf);
60             if (strcmp(buf,"MAKE-SAME") == 0)
61             {
62                 scanf("%d%d%d",&pos,&tot,&c);

```

```

60         sp.Select(pos-1,pos+tot-2);
61         sp.root->c[1]->c[0]->same = true;
62         sp.root->c[1]->c[0]->lazy = c;
63         sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
64     }
65     else if (strcmp(buf,"INSERT") == 0)
66     {
67         scanf("%d%d",&pos,&tot);
68         for (int i = 0;i < tot;i++)
69             scanf("%d",&a[i]);
70         sp.Insert(pos,a,tot);
71     }
72     else if (strcmp(buf,"DELETE") == 0)
73     {
74         scanf("%d%d",&pos,&tot);
75         sp.Remove(pos-1,pos+tot-2);
76     }
77     else if (strcmp(buf,"REVERSE") == 0)
78     {
79         scanf("%d%d",&pos,&tot);
80         sp.Select(pos-1,pos+tot-2);
81         sp.root->c[1]->c[0]->rev ^= 1;
82         sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
83     }
84     else if (strcmp(buf,"GET-SUM") == 0)
85     {
86         scanf("%d%d",&pos,&tot);
87         sp.Select(pos-1,pos+tot-2);
88         printf("%d\n",sp.root->c[1]->c[0]->sum);
89     }
90     else if (strcmp(buf,"MAX-SUM") == 0)
91     {
92         sp.Select(0,sp.root->size-3);
93         printf("%d\n",sp.root->c[1]->c[0]->maxsum);
94     }
95 }
96 }
97 return 0;
98 }

```

维护多个序列的时候，不需要建立很多Splay。只需要记录某个点在内存池中的绝对位置就可以了。需要操作它所在的序列时直接Splay到nil。此时Splay的root所在的Splay就是这个序列了。新建序列的时候需要多加入两个额外节点。如果某个Splay只有两个节点了需要及时回收。  
 例题：Box（维护括号序列）

```

1  \\\下面都是专用函数
2  \\\判断x在不在f里面
3  bool Ancestor(Node *x,Node *f)
4  {
5      if (x == f) return true;
6      while (x->p != nil)
7      {
8          if (x->p == f) return true;
9          x = x->p;
10     }
11     return false;
12 }
13 \\\把Splay v插入到pos后面, pos=nil时新开一个序列
14 void Insert(Node *pos, Node *v)
15 {
16     int pl;
17     if (pos == nil)
18     {

```

```

19         Init();
20         Insert(0), Insert(0);
21         pl = 0;
22     }
23     else
24     {
25         Splay(pos, nil);
26         pl = root->c[0]->size;
27     }
28     Select(pl, nil), Select(pl + 1, root);
29     root->c[1]->c[0] = v;
30     v->p = root->c[1];
31     Splay(v, nil);
32 }
33 \\把[l,r]转出来（这里记录的是绝对位置）
34 void Select(Node *l, Node *r)
35 {
36     Splay(l, nil);
37     int pl = root->c[0]->size - 1;
38     Splay(r, nil);
39     int pr = root->c[0]->size - 1;
40     Select(pl, pr);
41 }
42 \\分离[l,r]
43 Node *Split(Node *l, Node *r)
44 {
45     Select(l, r);
46     Node *res = root->c[1]->c[0];
47     root->c[1]->c[0] = res->p = nil;
48     Splay(root->c[1], nil);
49     if (root->size == 2)
50     {
51         Recycle(root);
52         Init();
53     }
54     return res;
55 }
56
57 int main(int argc, char const *argv[])
58 {
59     freopen("P.in", "r", stdin);
60     bool first = true;
61     while (scanf("%d", &n) != EOF)
62     {
63         if (!first) puts("");
64         first = false;
65         Init();
66         for (int i = 0; i < n; i++)
67         {
68             \\建立独立的N个区间，记录绝对位置
69             sp.Init();
70             sp.Insert(0), sp.Insert(0);
71             sp.Insert(0,i+1),sp.Insert(1,i+1);
72             sp.Select(0, 0), l[i] = sp.root->c[1]->c[0];
73             sp.Select(1, 1), r[i] = sp.root->c[1]->c[0];
74         }
75         for (int i = 0; i < n; i++)
76         {
77             int f;
78             scanf("%d", &f);
79             if (f != 0)
80             {
81                 \\把[l[i],r[i]]插入到l[f-1]后面

```



```

82         Node *pos = sp.Split(l[i], r[i]);
83         sp.Insert(l[f - 1], pos);
84     }
85 }
86 scanf("%d", &n);
87 for (int i = 0; i < n; i++)
88 {
89     scanf("%s", com);
90     if (com[0] == 'Q')
91     {
92         int pos;
93         scanf("%d", &pos);
94         \\求[l[pos-1],r[pos-1]]在哪个序列里面
95         sp.Splay(l[pos - 1], nil);
96         sp.Select(1, nil);
97         printf("%d\n", sp.root->key);
98     }
99     else
100     {
101         int u, v;
102         scanf("%d%d", &u, &v);
103         if (v == 0)
104             sp.Insert(nil, sp.Split(l[u-1], r[u-1]));
105         else
106         {
107             sp.Select(l[u-1], r[u-1]);
108             if (sp.Ancestor(l[v-1], sp.root->c[1]->c[0]) == false)
109                 sp.Insert(l[v - 1], sp.Split(l[u-1], r[u-1]));
110         }
111     }
112 }
113 }
114 return 0;
115 }

```

## 5.2 \*动态树

### 5.2.1 维护点权

被注释的部分是具体题目用到的东西。

支持换根。

Cut操作还没写。

```

1  const int MaxN = 110000;
2
3  struct Node
4  {
5      int size, key;
6      bool rev;
7
8      // bool same;
9      // int lsum, rsum, sum, maxsum, sa;
10
11      Node *c[2];
12      Node *p;
13 } mem[MaxN], *cur, *nil, *pos[MaxN];
14
15 Node *newNode(int v, Node *p)
16 {
17     cur->c[0] = cur->c[1] = nil, cur->p = p;
18     cur->size = 1;
19     cur->key = v;
20     cur->rev = false;

```

```

21
22 // cur->same = false;
23 // cur->sa = 0;
24 // cur->lsum = cur->rsum = cur->maxsum = 0;
25 // cur->sum = v;
26
27     return cur++;
28 }
29
30 void Init()
31 {
32     cur = mem;
33     nil = newNode(0, cur);
34     nil->size = 0;
35 }
36
37 struct SplayTree
38 {
39     void Pushup(Node *x)
40     {
41         if (x == nil) return;
42         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
43         x->size = x->c[0]->size + x->c[1]->size + 1;
44
45         // x->sum = x->c[0]->sum + x->c[1]->sum + x->key;
46         // x->lsum = max(x->c[0]->lsum, x->c[0]->sum + x->key + max(0, x->c
47         // x->rsum = max(x->c[1]->rsum, x->c[1]->sum + x->key + max(0, x->c
48         // x->maxsum = max(max(x->c[0]->maxsum, x->c[1]->maxsum),
49         // x->key + max(0, x->c[0]->rsum) + max(0, x->c[1]->lsum));
50
51     }
52     void Pushdown(Node *x)
53     {
54         if (x == nil) return;
55         if (x->rev)
56         {
57             x->rev = 0;
58             x->c[0]->rev ^= 1;
59             x->c[1]->rev ^= 1;
60             swap(x->c[0], x->c[1]);
61         //注意修改与位置有关的量
62         // swap(x->lsum, x->rsum);
63     }
64
65     // if (x->same)
66     // {
67     //     x->same = false;
68     //     x->key = x->sa;
69     //     x->sum = x->sa * x->size;
70     //     x->lsum = x->rsum = x->maxsum = max(0, x->sum);
71     //     if (x->c[0] != nil)
72     //         x->c[0]->same = true, x->c[0]->sa = x->sa;
73     //     if (x->c[1] != nil)
74     //         x->c[1]->same = true, x->c[1]->sa = x->sa;
75     // }
76 }
77 bool isRoot(Node *x)
78 {
79     return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
80 }
81 void Rotate(Node *x, int f)

```

```

82     {
83         if (isRoot(x))    return;
84         Node *y = x->p;
85         y->c[f ^ 1] = x->c[f], x->p = y->p;
86         if (x->c[f] != nil)
87             x->c[f]->p = y;
88         if (y != nil)
89         {
90             if (y == y->p->c[1])
91                 y->p->c[1] = x;
92             else if (y == y->p->c[0])
93                 y->p->c[0] = x;
94         }
95         x->c[f] = y, y->p = x;
96         Pushup(y);
97     }
98 void Splay(Node *x)
99 {
100     static Node *stack[MaxN];
101     int top = 0;
102     stack[top++] = x;
103     for (Node *y = x; !isRoot(y); y = y->p)
104         stack[top++] = y->p;
105     while (top)
106         Pushdown(stack[--top]);
107
108     while (!isRoot(x))
109     {
110         Node *y = x->p;
111         if (isRoot(y))
112             Rotate(x, x == y->c[0]);
113         else
114         {
115             int fd = y->p->c[0] == y;
116             if (y->c[fd] == x)
117                 Rotate(x, fd ^ 1), Rotate(x, fd);
118             else
119                 Rotate(y, fd), Rotate(x, fd);
120         }
121     }
122     Pushup(x);
123 }
124 Node *Access(Node *u)
125 {
126     Node *v = nil;
127     while (u != nil)
128     {
129         Splay(u);
130         v->p = u;
131         u->c[1] = v;
132         Pushup(u);
133         u = (v = u)->p;
134         if (u == nil)
135             return v;
136     }
137 }
138 Node *LCA(Node *u, Node *v)
139 {
140     Access(u);
141     return Access(v);
142 }
143 Node *Link(Node *u, Node *v)
144 {

```

```

145     Access(u);
146     Splay(u);
147     u->rev = true;
148     u->p = v;
149 }
150 void ChangeRoot(Node *u)
151 {
152     Access(u)->rev ^= 1;
153 }
154 Node *GetRoute(Node *u, Node *v)
155 {
156     ChangeRoot(u);
157     return Access(v);
158 }
159 };
160
161 int n, m;
162 SplayTree sp;
163
164 int main(int argc, char const *argv[])
165 {
166     while (scanf("%d", &n) != EOF)
167     {
168         Init();
169         for (int i = 0; i < n; i++)
170         {
171             int v;
172             scanf("%d", &v);
173             pos[i] = newNode(v, nil);
174         }
175         for (int i = 0; i < n - 1; i++)
176         {
177             int u, v;
178             scanf("%d%d", &u, &v);
179             u--, v--;
180             sp.Link(pos[u], pos[v]);
181         }
182
183         //     scanf("%d", &m);
184         //     for (int i = 0; i < m; i++)
185         //     {
186         //         int typ, u, v, c;
187         //         scanf("%d%d%d", &typ, &u, &v);
188         //         u--, v--;
189         //         if (typ == 1)
190         //             printf("%d\n", sp.GetRoute(pos[u], pos[v])->maxsum);
191         //         else
192         //         {
193         //             scanf("%d", &c);
194         //             Node *p = sp.GetRoute(pos[u], pos[v]);
195         //             p->same = true;
196         //             p->sa = c;
197         //         }
198         //     }
199     }
200     return 0;
201 }

```

### 5.3 可持久化线段树

区间第 $k$ 小数，内存压缩版，POJ2014。

```
1 #include <cstdio>
```

```

2  #include <algorithm>
3  using namespace std;
4
5  const int MAXN=100000,MAXM=100000;
6
7  struct node
8  {
9      node *l,*r;
10     int sum;
11 }tree[MAXN*4+MAXM*20];
12
13 int N;
14 node *newnode()
15 {
16     tree[N].l=tree[N].r=NULL;
17     tree[N].sum=0;
18     return &tree[N++];
19 }
20 node *newnode(node *x)
21 {
22     tree[N].l=x->l;
23     tree[N].r=x->r;
24     tree[N].sum=x->sum;
25     return &tree[N++];
26 }
27 node *build(int l,int r)
28 {
29     node *x=newnode();
30     if (l<r)
31     {
32         int mid=l+r>>1;
33         x->l=build(l,mid);
34         x->r=build(mid+1,r);
35         x->sum=x->l->sum+x->r->sum;
36     }
37     else
38         x->sum=0;
39     return x;
40 }
41 node *update(node *x,int l,int r,int p,int v)
42 {
43     if (l<r)
44     {
45         int mid=l+r>>1;
46         node *nx=newnode(x);
47         if (p<=mid)
48         {
49             node *ret=update(x->l,l,mid,p,v);
50             nx->l=ret;
51         }
52         else
53         {
54             node *ret=update(x->r,mid+1,r,p,v);
55             nx->r=ret;
56         }
57         nx->sum=nx->l->sum+nx->r->sum;
58         return nx;
59     }
60     else
61     {
62         node *nx=newnode(x);
63         nx->sum+=v;
64         return nx;

```

```

65     }
66 }
67 int query(node *x1,node *x2,int l,int r,int k)
68 {
69     if (l<r)
70     {
71         int mid=l+r>>1;
72         int lsum=x2->l->sum-x1->l->sum;
73         if (lsum>=k)
74             return query(x1->l,x2->l,l,mid,k);
75         else
76             return query(x1->r,x2->r,mid+1,r,k-lsum);
77     }
78     else
79         return l;
80 }
81 char s[10];
82 node *root[MAXM+1];
83 int a[MAXN],b[MAXN];
84 int init(int n)
85 {
86     for (int i=0;i<n;i++)
87         b[i]=a[i];
88     sort(b,b+n);
89     int tn=unique(b,b+n)-b;
90     for (int i=0;i<n;i++)
91     {
92         int l=0,r=tn-1;
93         while (l<r)
94         {
95             int mid=l+r>>1;
96             if (b[mid]>=a[i])
97                 r=mid;
98             else
99                 l=mid+1;
100         }
101         a[i]=l;
102     }
103     return tn;
104 }
105 int main()
106 {
107     int cas=1,n;
108     while (scanf("%d",&n)!=EOF)
109     {
110         printf("Case_␣%d:\n",cas++);
111         for (int i=0;i<n;i++)
112             scanf("%d",&a[i]);
113         int tn=init(n);
114         N=0;
115         root[0]=build(0,tn-1);
116         for (int i=1;i<=n;i++)
117             root[i]=update(root[i-1],0,tn-1,a[i-1],1);
118         int m;
119         scanf("%d",&m);
120         for (int i=0;i<m;i++)
121         {
122             int s,t;
123             scanf("%d%d",&s,&t);
124             printf("%d\n",b[query(root[s-1],root[t],0,tn-1,t-s+2>>1)]);
125         }
126     }
127     return 0;

```

## 5.4 treap正式版

支持翻转。

```

1 #include <cstdio>
2 #include <cstdlib>
3 #include <algorithm>
4 using namespace std;
5
6 const int MAXN = 100000;
7 const int MAXM = 100000;
8 const int inf = 0x7fffffff;
9 int a[MAXN];
10 struct Treap
11 {
12     int N;
13     Treap()
14     {
15         N = 0;
16         root = NULL;
17     }
18     void init()
19     {
20         N = 0;
21         root = NULL;
22     }
23     struct Treap_Node
24     {
25         Treap_Node *son[2]; //left & right
26         int value, fix;
27         bool lazy;
28         int size;
29         Treap_Node() {}
30         Treap_Node(int _value)
31         {
32             son[0] = son[1] = NULL;
33             value = _value;
34             fix = rand() * rand();
35             lazy = 0;
36             size = 1;
37         }
38         int sonSize(bool flag)
39         {
40             if (son[flag] == NULL)
41                 return 0;
42             else
43                 return son[flag]->size;
44         }
45     } node[MAXN], *root, *pos[MAXN];
46     void up(Treap_Node *p)
47     {
48         p->size = p->sonSize(0) + p->sonSize(1) + 1;
49     }
50     void down(Treap_Node *p)
51     {
52         if (!p->lazy)
53             return ;
54         for (int i = 0; i < 2; i++)
55             if (p->son[i])
56                 p->son[i]->lazy = !p->son[i]->lazy;
57         swap(p->son[0], p->son[1]);

```

```

58     p->lazy = 0;
59 }
60 Treap_Node *merge(Treap_Node *p, Treap_Node *q)
61 {
62     if (p == NULL)
63         return q;
64     else if (q == NULL)
65         return p;
66     if (p->fix <= q->fix)
67     {
68         down(p);
69         p->son[1] = merge(p->son[1], q);
70         up(p);
71         return p;
72     }
73     else
74     {
75         down(q);
76         q->son[0] = merge(p, q->son[0]);
77         up(q);
78         return q;
79     }
80 }
81 pair<Treap_Node *, Treap_Node *> split(Treap_Node *p, int n)
82 {
83     if (p == NULL)
84         return make_pair((Treap_Node *)NULL, (Treap_Node *)NULL);
85     if (!n)
86         return make_pair((Treap_Node *)NULL, p);
87     if (n == p->size)
88         return make_pair(p, (Treap_Node *)NULL);
89     down(p);
90     if (p->sonSize(0) >= n)
91     {
92         pair<Treap_Node *, Treap_Node *> ret = split(p->son[0], n);
93         p->son[0] = ret.second;
94         up(p);
95         return make_pair(ret.first, p);
96     }
97     else
98     {
99         pair<Treap_Node *, Treap_Node *> ret = split(p->son[1], n - p->
100             sonSize(0) - 1);
101         p->son[1] = ret.first;
102         up(p);
103         return make_pair(p, ret.second);
104     }
105 }
106 int smalls(Treap_Node *p, int value)
107 {
108     if (p==NULL)
109         return 0;
110     if (p->value<=value)
111         return 1+p->sonSize(0)+smalls(p->son[1], value);
112     else
113         return smalls(p->son[0], value);
114 }
115 void insert(int value)
116 {
117     Treap_Node *p = &node[N++];
118     *p = Treap_Node(value);
119     pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root,
120         value));

```



```

119         root = merge(merge(ret.first, p), ret.second);
120     }
121     void remove(int value)
122     {
123         pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root,
124             value) - 1);
125         root = merge(ret.first, split(ret.second, 1).second);
126     }
127     Treap_Node *build(int s, int t)
128     {
129         int idx = t + s >> 1;
130         Treap_Node *p = &node[N++];
131         *p = Treap_Node(a[idx]);
132         pos[a[idx]] = p;
133         if (idx > s)
134             p = merge(build(s, idx - 1), p);
135         if (idx < t)
136             p = merge(p, build(idx + 1, t));
137         up(p);
138         return p;
139     }
140     void build(int n)
141     {
142         root = build(0, n - 1);
143     }
144     void *reverse(int s, int t)
145     {
146         pair<Treap_Node *, Treap_Node *> tmp1, tmp2;
147         tmp1 = split(root, s - 1);
148         tmp2 = split(tmp1.second, t - s + 1);
149         tmp2.first->lazy = !tmp2.first->lazy;
150         root = merge(tmp1.first, merge(tmp2.first, tmp2.second));
151     }
152 };
153 Treap treap;
154 int main()
155 {
156     treap.init();
157     int n;
158     scanf("%d", &n);
159     for (int i = 0; i < n; i++)
160         scanf("%d", &a[i]);
161     treap.build(n);
162 }

```

## 5.5 树链剖分

### 5.5.1 点权

```

1  #include <cstdio>
2  #include <cstring>
3  #include <cstdlib>
4  #include <algorithm>
5  using namespace std;
6  const int MAX = 12000;
7  const int LOG = 15;
8  const int oo = 0x3f3f3f3f;
9  struct Edge
10 {
11     int to, w, id;
12     Edge* next;
13 } memo[MAX << 1], *cur, *g[MAX], *pree[MAX], *solid[MAX], *valid[MAX];
14 int dp[MAX][LOG], pos[MAX], lst[MAX], dep[MAX], cnt[MAX], h[MAX], K, n;

```

```

15 void init()
16 {
17     for (int i = 1; i <= n; i++)
18     {
19         g[i] = NULL;
20         valid[i] = NULL;
21         solid[i] = NULL;
22         pree[i] = NULL;
23     }
24     for (int i = 0; i < LOG; i++)
25     {
26         dp[1][i] = 1;
27     }
28     cur = memo;
29     K = 0;
30 }
31 void add(int u, int v, int w, int id)
32 {
33     cur->to = v;
34     cur->w = w;
35     cur->id = id;
36     cur->next = g[u];
37     g[u] = cur++;
38 }
39 void dfsLCA(int d, int u, int f)
40 {
41     dep[u] = d;
42     dp[u][0] = f;
43     cnt[u] = 1;
44     for (int i = 1; i < LOG; i++)
45     {
46         dp[u][i] = dp[dp[u][i - 1]][i - 1];
47     }
48     for (Edge* it = g[u]; it; it = it->next)
49     {
50         int v = it->to;
51         if (v != f)
52         {
53             pree[v] = it;
54             valid[it->id] = it;
55             dfsLCA(d + 1, v, u); //RE
56             cnt[u] += cnt[v];
57             if (solid[u] == NULL || cnt[solid[u]->to] < cnt[v])
58             {
59                 solid[u] = it;
60             }
61         }
62     }
63 }
64 void dfsChain(int u, int head)
65 {
66     h[u] = head;
67     if (solid[u])
68     {
69         lst[pos[u] = K++] = u;
70         dfsChain(solid[u]->to, head);
71     }
72     else
73     for (Edge* it = g[u]; it; it = it->next)
74     {
75         int v = it->to;
76         if (it != solid[u] && v != dp[u][0])
77         {

```

```

78         dfsChain(v, v);
79     }
80 }
81 }
82 int getLCA(int u, int v)
83 {
84     if (dep[u] < dep[v])
85         swap(u, v);
86     for (int st = 1 << (LOG - 1), i = LOG - 1; i >= 0; i--, st >>= 1)
87     {
88         if (st <= dep[u] - dep[v])
89         {
90             u = dp[u][i];
91         }
92     }
93     if (u == v)
94         return u;
95     for (int i = LOG - 1; i >= 0; i--)
96     {
97         if (dp[u][i] != dp[v][i])
98         {
99             u = dp[u][i];
100            v = dp[v][i];
101        }
102    }
103    return dp[u][0];
104 }
105 struct Node
106 {
107     int l, r, ma, mi;
108     bool rev;
109 } seg[MAX << 2];
110 void reverse(int k)
111 {
112     seg[k].mi *= -1;
113     seg[k].ma *= -1;
114     seg[k].rev ^= 1;
115     swap(seg[k].mi, seg[k].ma);
116 }
117 void pushdown(int k)
118 {
119     if (seg[k].rev)
120     {
121         reverse(k << 1);
122         reverse(k << 1 | 1);
123         seg[k].rev = false;
124     }
125 }
126 void update(int k)
127 {
128     seg[k].mi = min(seg[k << 1].mi, seg[k << 1 | 1].mi);
129     seg[k].ma = max(seg[k << 1].ma, seg[k << 1 | 1].ma);
130 }
131 void init(int k, int l, int r)
132 {
133     seg[k].l = l;
134     seg[k].r = r;
135     seg[k].rev = false;
136     if (l == r)
137     {
138         seg[k].mi = seg[k].ma = solid[lst[l]]->w; //solid WA
139         return;
140     }

```

```

141     int mid = l + r >> 1;
142     init(k << 1, l, mid);
143     init(k << 1 | 1, mid + 1, r);
144     update(k);
145 }
146 void update(int k, int id, int v)
147 {
148     if (seg[k].l == seg[k].r)
149     {
150         seg[k].mi = seg[k].ma = solid[lst[id]]->w = v;
151         return;
152     }
153     pushdown(k);
154     int mid = seg[k].l + seg[k].r >> 1;
155     if (id <= mid)
156         update(k << 1, id, v);
157     else
158         update(k << 1 | 1, id, v);
159     update(k);
160 }
161 void reverse(int k, int l, int r)
162 {
163     if (seg[k].l > r || seg[k].r < l)
164         return;
165     if (seg[k].l >= l && seg[k].r <= r)
166     {
167         reverse(k);
168         return;
169     }
170     pushdown(k);
171     reverse(k << 1, l, r);
172     reverse(k << 1 | 1, l, r);
173     update(k);
174 }
175 int read(int k, int l, int r)
176 {
177     if (seg[k].l > r || seg[k].r < l)
178         return -oo;
179     if (seg[k].l >= l && seg[k].r <= r)
180         return seg[k].ma;
181     pushdown(k);
182     return max(read(k << 1, l, r), read(k << 1 | 1, l, r));
183 }
184 void setEdge(int id, int v)
185 {
186     Edge* it = valid[id];
187     if (h[it->to] != it->to)
188     {
189         update(1, pos[dp[it->to][0]], v);
190     }
191     else
192     {
193         it->w = v;
194     }
195 }
196 void negateLCA(int t, int u)
197 {
198     while (t != u)
199     {
200         int tmp = h[u];
201         if (dep[tmp] < dep[t])
202             tmp = t;
203         if (h[u] == u)

```

```

204     {
205         pree[u]->w *= -1;
206         u = dp[u][0];
207     }
208     else
209     {
210         reverse(1, pos[tmp], pos[dp[u][0]]);
211         u = tmp;
212     }
213 }
214 }
215 void negate(int u, int v)
216 {
217     int t = getLCA(u, v);
218     negateLCA(t, u);
219     negateLCA(t, v);
220 }
221 int maxLCA(int t, int u)
222 {
223     int ret = -oo;
224     while (t != u)
225     {
226         int tmp = h[u];
227         if (dep[tmp] < dep[t])
228             tmp = t;
229         if (h[u] == u)
230         {
231             ret = max(ret, pree[u]->w);
232             u = dp[u][0];
233         }
234         else
235         {
236             ret = max(ret, read(1, pos[tmp], pos[dp[u][0]]));
237             u = tmp;
238         }
239     }
240     return ret;
241 }
242 int query(int u, int v)
243 {
244     int t = getLCA(u, v);
245     return max(maxLCA(t, u), maxLCA(t, v));
246 }
247 int main()
248 {
249     int T;
250     int u, v, w;
251     char op[15];
252     scanf("%d", &T);
253     while (T--)
254     {
255         scanf("%d", &n);
256         init();
257         for (int i = 1; i < n; i++)
258         {
259             scanf("%d%d%d", &u, &v, &w);
260             add(u, v, w, i);
261             add(v, u, w, i);
262         }
263         dfsLCA(0, 1, 1);
264         dfsChain(1, 1);
265         init(1, 0, K - 1);
266         while (scanf("%s", op), op[0] != 'D')

```

```

267     {
268         scanf("%d%d", &u, &v);
269         if (op[0] == 'C')
270         {
271             setEdge(u, v);
272         }
273         else if (op[0] == 'N')
274         {
275             negate(u, v);
276         }
277         else
278         {
279             printf("%d\n", query(u, v));
280         }
281     }
282 }
283 return 0;
284 }

```

### 5.5.2 边权

```

1  #include <cstdio>
2  #include <iostream>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <cmath>
6  #include <cstring>
7  using namespace std;
8  int n,m,sum,pos;
9  int head[50005],e;
10 int s[50005],from[50005];
11 int fa[50005][20],deep[50005],num[50005];
12 int solid[50005],p[50005],fp[50005];
13 struct N
14 {
15     int l,r,mid;
16     int add,w;
17 }nod[50005*4];
18 struct M
19 {
20     int v,next;
21 }edge[100005];
22 void addedge(int u,int v)
23 {
24     edge[e].v=v;
25     edge[e].next=head[u];
26     head[u]=e++;
27
28     edge[e].v=u;
29     edge[e].next=head[v];
30     head[v]=e++;
31 }
32 void LCA(int st,int f,int d)
33 {
34     deep[st]=d;
35     fa[st][0]=f;
36     num[st]=1;
37     int i,v;
38     for(i=1;i<20;i++)
39         fa[st][i]=fa[fa[st][i-1]][i-1];
40     for(i=head[st];i!=-1;i=edge[i].next)
41     {
42         v=edge[i].v;
43         if(v!=f)

```

```

44         {
45             LCA(v,st,d+1);
46             num[st]+=num[v];
47             if(solid[st]==-1||num[v]>num[solid[st]])
48                 solid[st]=v;
49         }
50     }
51 }
52 void getpos(int st,int sp)
53 {
54     from[st]=sp;
55     if(solid[st]!=-1)
56     {
57         p[st]=pos++;
58         fp[p[st]]=st;
59         getpos(solid[st],sp);
60     }
61     else
62     {
63         p[st]=pos++;
64         fp[p[st]]=st;
65         return;
66     }
67     int i,v;
68     for(i=head[st];i!=-1;i=edge[i].next)
69     {
70         v=edge[i].v;
71         if(v!=solid[st]&&v!=fa[st][0])
72             getpos(v,v);
73     }
74 }
75 int getLCA(int u,int v)
76 {
77     if(deep[u]<deep[v])
78         swap(u,v);
79     int d=1<<19,i;
80     for(i=19;i>=0;i--)
81     {
82         if(d<=deep[u]-deep[v])
83             u=fa[u][i];
84         d>>=1;
85     }
86     if(u==v)
87         return u;
88     for(i=19;i>=0;i--)
89         if(fa[u][i]!=fa[v][i])
90         {
91             u=fa[u][i];
92             v=fa[v][i];
93         }
94     return fa[u][0];
95 }
96 void init(int p,int l,int r)
97 {
98     nod[p].l=l;
99     nod[p].r=r;
100     nod[p].mid=(l+r)>>1;
101     nod[p].add=0;
102     if(l==r)
103         nod[p].w=s[fp[l]];
104     else
105     {
106         init(p<<1,l,nod[p].mid);

```

```

107         init(p<<1|1,nod[p].mid+1,r);
108     }
109 }
110 void lazy(int p)
111 {
112     if(nod[p].add!=0)
113     {
114         nod[p<<1].add+=nod[p].add;
115         nod[p<<1|1].add+=nod[p].add;
116         nod[p].add=0;
117     }
118 }
119 void update(int p,int l,int r,int v)
120 {
121     if(nod[p].l==l&&nod[p].r==r)
122     {
123         nod[p].add+=v;
124         return;
125     }
126     lazy(p);
127     if(nod[p].mid<l)
128         update(p<<1|1,l,r,v);
129     else if(nod[p].mid>=r)
130         update(p<<1,l,r,v);
131     else
132     {
133         update(p<<1,l,nod[p].mid,v);
134         update(p<<1|1,nod[p].mid+1,r,v);
135     }
136 }
137 int read(int p,int l,int r)
138 {
139     if(nod[p].l==l&&nod[p].r==r)
140         return nod[p].w+nod[p].add;
141     lazy(p);
142     if(nod[p].mid<l)
143         return read(p<<1|1,l,r);
144     else if(nod[p].mid>=r)
145         return read(p<<1,l,r);
146 }
147 void jump(int st,int ed,int val)
148 {
149     while(deep[st]>=deep[ed])
150     {
151         int tmp=from[st];
152         if(deep[tmp]<deep[ed])
153             tmp=ed;
154         update(1,p[tmp],p[st],val);
155         st=fa[tmp][0];
156     }
157 }
158 void change(int st,int ed,int val)
159 {
160     int lca=getLCA(st,ed);
161     jump(st,lca,val);
162     jump(ed,lca,val);
163     jump(lca,lca,-val);
164 }
165 int main()
166 {
167     while(scanf("%d%d%d",&n,&m,&sum)==3)
168     {
169         int i;

```



```

170     s[0]=0;pos=0;deep[0]=-1;
171     memset(fa,0,sizeof(fa));
172     for(i=1;i<=n;i++)
173     {
174         solid[i]=-1;
175         scanf("%d",&s[i]);
176     }
177     memset(head,-1,sizeof(head));
178     e=0;
179     for(i=0;i<m;i++)
180     {
181         int a,b;
182         scanf("%d%d",&a,&b);
183         addedge(a,b);
184     }
185     LCA(1,0,0);
186     getpos(1,1);
187     init(1,0,pos-1);
188     for(i=0;i<sum;i++)
189     {
190         char que[5];
191         scanf("%s",que);
192         if(que[0]!='Q')
193         {
194             int a,b,c;
195             scanf("%d%d%d",&a,&b,&c);
196             if(que[0]=='D')
197                 c=-c;
198             change(a,b,c);
199         }
200         else
201         {
202             int a;
203             scanf("%d",&a);
204             printf("%d\n",read(1,p[a],p[a]));
205         }
206     }
207 }
208 return 0;
209 }

```

## 6 图论

### 6.1 SAP四版

```
1  const int MAXEDGE=20400;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  struct edges
5  {
6      int cap,to,next,flow;
7  } edge[MAXEDGE+100];
8  struct nodes
9  {
10     int head,label,pre,cur;
11 } node[MAXN+100];
12 int L,N;
13 int gap[MAXN+100];
14 void init(int n)
15 {
16     L=0;
17     N=n;
18     for (int i=0; i<N; i++)
19         node[i].head=-1;
20 }
21 void add_edge(int x,int y,int z,int w)
22 {
23     edge[L].cap=z;
24     edge[L].flow=0;
25     edge[L].to=y;
26     edge[L].next=node[x].head;
27     node[x].head=L++;
28     edge[L].cap=w;
29     edge[L].flow=0;
30     edge[L].to=x;
31     edge[L].next=node[y].head;
32     node[y].head=L++;
33 }
34 int maxflow(int s,int t)
35 {
36     memset(gap,0,sizeof(gap));
37     gap[0]=N;
38     int u,ans=0;
39     for (int i=0; i<N; i++)
40     {
41         node[i].cur=node[i].head;
42         node[i].label=0;
43     }
44     u=s;
45     node[u].pre=-1;
46     while (node[s].label<N)
47     {
48         if (u==t)
49         {
50             int min=inf;
51             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
52                 if (min>edge[i].cap-edge[i].flow)
53                     min=edge[i].cap-edge[i].flow;
54             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
55             {
56                 edge[i].flow+=min;
57                 edge[i^1].flow-=min;
58             }
59             u=s;
```

```

60         ans+=min;
61         continue;
62     }
63     bool flag=false;
64     int v;
65     for (int i=node[u].cur; i!=-1; i=edge[i].next)
66     {
67         v=edge[i].to;
68         if (edge[i].cap-edge[i].flow && node[v].label+1==node[u].label)
69         {
70             flag=true;
71             node[u].cur=node[v].pre=i;
72             break;
73         }
74     }
75     if (flag)
76     {
77         u=v;
78         continue;
79     }
80     node[u].cur=node[u].head;
81     int min=N;
82     for (int i=node[u].head; i!=-1; i=edge[i].next)
83         if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
84             min=node[edge[i].to].label;
85     gap[node[u].label]--;
86     if (!gap[node[u].label]) return ans;
87     node[u].label=min+1;
88     gap[node[u].label]++;
89     if (u!=s) u=edge[node[u].pre^1].to;
90 }
91 return ans;
92 }

```

## 6.2 费用流三版

T了可以改成栈。

```

1  const int MAXM=60000;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  int L,N;
5  int K;
6  struct edges
7  {
8      int to,next,cap,flow,cost;
9  } edge[MAXM];
10 struct nodes
11 {
12     int dis,pre,head;
13     bool visit;
14 } node[MAXN];
15 void init(int n)
16 {
17     N=n;
18     L=0;
19     for (int i=0; i<N; i++)
20         node[i].head=-1;
21 }
22 void add_edge(int x,int y,int cap,int cost)
23 {
24     edge[L].to=y;
25     edge[L].cap=cap;
26     edge[L].cost=cost;

```

```

27     edge[L].flow=0;
28     edge[L].next=node[x].head;
29     node[x].head=L++;
30     edge[L].to=x;
31     edge[L].cap=0;
32     edge[L].cost=-cost;
33     edge[L].flow=0;
34     edge[L].next=node[y].head;
35     node[y].head=L++;
36 }
37 bool spfa(int s,int t)
38 {
39     queue <int> q;
40     for (int i=0; i<N; i++)
41     {
42         node[i].dis=0x3fffffff;
43         node[i].pre=-1;
44         node[i].visit=0;
45     }
46     node[s].dis=0;
47     node[s].visit=1;
48     q.push(s);
49     while (!q.empty())
50     {
51         int u=q.front();
52         node[u].visit=0;
53         for (int i=node[u].head; i!=-1; i=edge[i].next)
54         {
55             int v=edge[i].to;
56             if (edge[i].cap>edge[i].flow &&
57                 node[v].dis>node[u].dis+edge[i].cost)
58             {
59                 node[v].dis=node[u].dis+edge[i].cost;
60                 node[v].pre=i;
61                 if (!node[v].visit)
62                 {
63                     node[v].visit=1;
64                     q.push(v);
65                 }
66             }
67         }
68         q.pop();
69     }
70     if (node[t].pre==-1)
71         return 0;
72     else
73         return 1;
74 }
75 int mcmf(int s,int t,int &cost)
76 {
77     int flow=0;
78     while (spfa(s,t))
79     {
80         int max=inf;
81         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
82         {
83             if (max>edge[i].cap-edge[i].flow)
84                 max=edge[i].cap-edge[i].flow;
85         }
86         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
87         {
88             edge[i].flow+=max;
89             edge[i^1].flow-=max;

```

```

90         cost+=edge[i].cost*max;
91     }
92     flow+=max;
93 }
94 return flow;
95 }

```

### 6.3 一般图匹配带花树

```

1  const int MaxN = 222;
2  int N;
3  bool Graph[MaxN+1][MaxN+1];
4  int Match[MaxN+1];
5  bool InQueue[MaxN+1], InPath[MaxN+1], InBlossom[MaxN+1];
6  int Head, Tail;
7  int Queue[MaxN+1];
8  int Start, Finish;
9  int NewBase;
10 int Father[MaxN+1], Base[MaxN+1];
11 int Count;
12 void CreateGraph()
13 {
14     int u, v;
15     memset(Graph, false, sizeof(Graph));
16     scanf("%d", &N);
17     while (scanf("%d%d", &u, &v) != EOF)
18         Graph[u][v] = Graph[v][u] = true;
19 }
20 void Push(int u)
21 {
22     Queue[Tail] = u;
23     Tail++;
24     InQueue[u] = true;
25 }
26 int Pop()
27 {
28     int res = Queue[Head];
29     Head++;
30     return res;
31 }
32 int FindCommonAncestor(int u, int v)
33 {
34     memset(InPath, false, sizeof(InPath));
35     while (true)
36     {
37         u = Base[u];
38         InPath[u] = true;
39         if (u == Start) break;
40         u = Father[Match[u]];
41     }
42     while (true)
43     {
44         v = Base[v];
45         if (InPath[v]) break;
46         v = Father[Match[v]];
47     }
48     return v;
49 }
50 void ResetTrace(int u)
51 {
52     int v;
53     while (Base[u] != NewBase)

```

```

54     {
55         v = Match[u];
56         InBlossom[Base[u]] = InBlossom[Base[v]] = true;
57         u = Father[v];
58         if (Base[u] != NewBase) Father[u] = v;
59     }
60 }
61 void BlossomContract(int u,int v)
62 {
63     NewBase = FindCommonAncestor(u,v);
64     memset(InBlossom,false,sizeof(InBlossom));
65     ResetTrace(u);
66     ResetTrace(v);
67     if (Base[u] != NewBase) Father[u] = v;
68     if (Base[v] != NewBase) Father[v] = u;
69     for (int tu = 1; tu <= N; tu++)
70         if (InBlossom[Base[tu]])
71             {
72                 Base[tu] = NewBase;
73                 if (!InQueue[tu]) Push(tu);
74             }
75 }
76 void FindAugmentingPath()
77 {
78     memset(InQueue,false,sizeof(InQueue));
79     memset(Father,0,sizeof(Father));
80     for (int i = 1; i <= N; i++)
81         Base[i] = i;
82     Head = Tail = 1;
83     Push(Start);
84     Finish = 0;
85     while (Head < Tail)
86     {
87         int u = Pop();
88         for (int v = 1; v <= N; v++)
89             if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v))
90                 {
91                     if ((v == Start) || ((Match[v] > 0) && (Father[Match[v]] >
92                         0)))
93                         BlossomContract(u,v);
94                     else if (Father[v] == 0)
95                     {
96                         Father[v] = u;
97                         if (Match[v] > 0)
98                             Push(Match[v]);
99                     }
100                     else
101                     {
102                         Finish = v;
103                         return;
104                     }
105                 }
106     }
107 void AugmentPath()
108 {
109     int u,v,w;
110     u = Finish;
111     while (u > 0)
112     {
113         v = Father[u];
114         w = Match[v];
115         Match[v] = u;

```

```

116         Match[u] = v;
117         u = w;
118     }
119 }
120 void Edmonds()
121 {
122     memset(Match,0,sizeof(Match));
123     for (int u = 1; u <= N; u++)
124         if (Match[u] == 0)
125         {
126             Start = u;
127             FindAugmentingPath();
128             if (Finish > 0) AugmentPath();
129         }
130 }
131 void PrintMatch()
132 {
133     for (int u = 1; u <= N; u++)
134         if (Match[u] > 0)
135             Count++;
136     printf("%d\n",Count);
137     for (int u = 1; u <= N; u++)
138         if (u < Match[u])
139             printf("%d□%d\n",u,Match[u]);
140 }
141 int main()
142 {
143     CreateGraph();
144     Edmonds();
145     PrintMatch();
146 }

```

## 6.4 \*二维平面图的最大流

待整理

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  #include <vector>
6  #include <cmath>
7  #include <map>
8  #include <queue>
9  using namespace std;
10
11 const int maxn = 100100;
12 const int inf = 0x3f3f3f3f;
13 struct Point
14 {
15     int x,y,id;
16     double theta;
17     Point() {}
18     Point(int _x,int _y)
19     {
20         x = _x;
21         y = _y;
22     }
23     Point(Point _s,Point _e,int _id)
24     {
25         id = _id;
26         x = _s.x-_e.x;
27         y = _s.y-_e.y;
28         theta = atan2(y,x);

```

```

29     }
30     bool operator < (const Point &b) const
31     {
32         return theta < b.theta;
33     }
34 };
35
36 map<pair<int,int>,int > idmap;
37 struct Edge
38 {
39     int from,to,next,cap,near,mark;
40 };
41 Edge edge[maxn*2];
42 int head[maxn],L;
43 int cntd[maxn];
44 void addedge(int u,int v,int cap)
45 {
46     cntd[u]++;
47     cntd[v]++;
48     idmap[make_pair(u,v)] = L;
49     edge[L].from = u;
50     edge[L].to = v;
51     edge[L].cap = cap;
52     edge[L].next = head[u];
53     edge[L].mark = -1;
54     head[u] = L++;
55 }
56
57 int rtp[maxn];
58 Point p[maxn],tp[maxn];
59 int n,m,S,T;
60 int vid;
61
62 struct Edge2
63 {
64     int to,next,dis;
65 } edge2[maxn*2];
66 int head2[maxn],L2;
67
68 void addedge2(int u,int v,int dis)
69 {
70     edge2[L2].to = v;
71     edge2[L2].dis = dis;
72     edge2[L2].next = head2[u];
73     head2[u] = L2++;
74 }
75
76 int dist[maxn];
77 bool inq[maxn];
78 int SPFA(int s,int t)
79 {
80     queue<int> Q;
81     memset(inq,false,sizeof(inq));
82     memset(dist,63,sizeof(dist));
83     Q.push(s);
84     dist[s] = 0;
85     while (!Q.empty())
86     {
87         int now = Q.front();
88         Q.pop();
89         for (int i = head2[now]; i != -1; i = edge2[i].next)
90             if (dist[edge2[i].to] > dist[now]+edge2[i].dis)
91                 {

```



```

92         dist[edge2[i].to] = dist[now]+edge2[i].dis;
93         if (inq[edge2[i].to] == false)
94         {
95             inq[edge2[i].to] = true;
96             Q.push(edge2[i].to);
97         }
98     }
99     inq[now] = false;
100 }
101 return dist[t];
102 }
103
104 int main()
105 {
106     int totcas;
107     scanf("%d",&totcas);
108     for (int cas = 1; cas <= totcas; cas++)
109     {
110         idmap.clear();
111         L = 0;
112         scanf("%d%d",&n,&m);
113         S = T = 0;
114         for (int i = 0; i < n; i++)
115         {
116             head[i] = -1;
117             scanf("%d%d",&p[i].x,&p[i].y);
118             if (p[S].x > p[i].x)
119                 S = i;
120             if (p[T].x < p[i].x)
121                 T = i;
122             cntd[i] = 0;
123         }
124         //源汇中间加入一个特殊节点
125         head[n] = -1;
126         n ++;
127         addedge(S,n-1,inf);
128         addedge(n-1,S,inf);
129         addedge(T,n-1,inf);
130         addedge(n-1,T,inf);
131
132         for (int i = 0; i < m; i++)
133         {
134             int u,v,cap;
135             scanf("%d%d%d",&u,&v,&cap);
136             u--;
137             v--;
138             addedge(u,v,cap);
139             addedge(v,u,cap);
140         }
141
142         for (int i = 0; i < n; i++)
143         {
144             int tot = 0;
145             //源点汇点连到特殊点的方向需要特别考虑一下
146             if (i == S)
147                 tp[tot++] = Point(Point(0,0),Point(-1,0),n-1);
148             else if (i == T)
149                 tp[tot++] = Point(Point(0,0),Point(1,0),n-1);
150             else if (i == n-1)
151             {
152                 tp[tot++] = Point(Point(0,0),Point(1,0),S);
153                 tp[tot++] = Point(Point(0,0),Point(-1,0),T);
154             }

```

```

155         if (i < n-1)
156         {
157             for (int j = head[i]; j != -1; j = edge[j].next)
158             {
159                 if (i == S && edge[j].to == n-1) continue;
160                 if (i == T && edge[j].to == n-1) continue;
161                 tp[tot++] = Point(p[i],p[edge[j].to],edge[j].to);
162             }
163         }
164         sort(tp,tp+tot);
165         for (int j = 0; j < tot; j++)
166             rtp[tp[j].id] = j;
167         for (int j = head[i]; j != -1; j = edge[j].next)
168             edge[j].near = tp[(rtp[edge[j].to]+1)%tot].id;
169     }
170
171     vid = 0;
172     for (int i = 0; i < L; i++)
173         if (edge[i].mark == -1)
174         {
175             int now = edge[i].from;
176             int eid = i;
177             int to = edge[i].to;
178             while (true)
179             {
180                 edge[eid].mark = vid;
181                 eid ^= 1;
182                 now = to;
183                 to = edge[eid].near;
184                 eid = idmap[make_pair(now,to)];
185
186                 if (now == edge[i].from) break;
187             }
188             vid++;
189         }
190
191     L2 = 0;
192     for (int i = 0; i < vid; i++)
193         head2[i] = -1;
194     for (int i = 0; i < L; i++)
195         addedge2(edge[i].mark,edge[i^1].mark,edge[i].cap);
196     printf("%d\n",SPFA(edge[0].mark,edge[1].mark));
197 }
198 return 0;
199 }

```

## 7 计算几何

太乱了尼玛。。

浮点数千万不要直接比较大小，千万要加上EPS啊混蛋。

### 7.1 基本函数

#### 7.1.1 Point定义

```
1 struct Point
2 {
3     double x, y;
4     Point() {}
5     Point(double _x, double _y)
6     {
7         x = _x, y = _y;
8     }
9     Point operator -(const Point &b) const
10    {
11        return Point(x - b.x, y - b.y);
12    }
13    double operator *(const Point &b) const
14    {
15        return x * b.y - y * b.x;
16    }
17    double operator &(const Point &b) const
18    {
19        return x * b.x + y * b.y;
20    }
21 };
```

#### 7.1.2 Line定义

```
1 struct Line
2 {
3     Point s, e;
4     double k;
5     Line() {}
6     Line(Point _s, Point _e)
7     {
8         s = _s, e = _e;
9         k = atan2(e.y - s.y, e.x - s.x);
10    }
11    Point operator &(const Line &b) const
12    {
13        Point res = s;
14        //注意：有些题目可能会有直线相交或者重合情况
15        //可以把返回值改成pair<Point,int>来返回两直线的状态。
16        double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
17        res.x += (e.x - s.x) * t;
18        res.y += (e.y - s.y) * t;
19        return res;
20    }
21 };
```

#### 7.1.3 距离：两点距离

```
1 double dist2(Point a, Point b)
2 {
3     return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
4 }
```

#### 7.1.4 距离：点到线段距离

res: 点到线段最近点

```

1 double dist2(Point p1, Point p2, Point p)
2 {
3     Point res;
4     double a, b, t;
5     a = p2.x - p1.x;
6     b = p2.y - p1.y;
7     t = ((p.x - p1.x) * a + (p.y - p1.y) * b) / (a * a + b * b);
8     if (t >= 0 && t <= 1)
9     {
10         res.x = p1.x + a * t;
11         res.y = p1.y + b * t;
12     }
13     else
14     {
15         if (dist2(p, p1) < dist2(p, p2))
16             res = p1;
17         else
18             res = p2;
19     }
20     return dist2(p, res);
21 }

```

### 7.1.5 面积: 多边形

点按逆时针排序。

```

1 double CalcArea(Point p[], int n)
2 {
3     double res = 0;
4     for (int i = 0; i < n; i++)
5         res += (p[i] * p[(i + 1) % n]) / 2;
6     return res;
7 }

```

### 7.1.6 判断: 线段相交

```

1 bool inter(Line l1, Line l2)
2 {
3     return (max(l1.s.x, l1.e.x) >= min(l2.s.x, l2.e.x) &&
4             max(l2.s.x, l2.e.x) >= min(l1.s.x, l1.e.x) &&
5             max(l1.s.y, l1.e.y) >= min(l2.s.y, l2.e.y) &&
6             max(l2.s.y, l2.e.y) >= min(l1.s.y, l1.e.y) &&
7             ((l2.s - l1.s) * (l1.e - l1.s)) * ((l2.e - l1.s) * (l1.e - l1.s)) <= 0 &&
8             ((l1.s - l2.s) * (l2.e - l2.s)) * ((l1.e - l2.s) * (l2.e - l2.s)) <= 0);
9 }

```

### 7.1.7 求解: 点到线最近点

```

1 Point NPT(Point P, Line L)
2 {
3     Point result;
4     double a, b, t;
5     a = L.e.x - L.s.x;
6     b = L.e.y - L.s.y;
7     t = ((P.x - L.s.x) * a + (P.y - L.s.y) * b) / (a * a + b * b);
8     //如果t小于0或者大于1,说明最近点在L.s和L.e这条线段之外
9     result.x = L.s.x + a * t;
10    result.y = L.s.y + b * t;
11    return result;
12 }

```

## 7.2 重心

```

1 Point CenterOfPolygon(Point poly[],int n)
2 {
3     Point p, p0, p1, p2, p3;
4     double m, m0;
5     p1 = poly[0];
6     p2 = poly[1];
7     p.x = p.y = m = 0;
8     for (int i = 2; i < n; i++)
9     {
10        p3 = poly[i];
11        p0.x = (p1.x + p2.x + p3.x) / 3.0;
12        p0.y = (p1.y + p2.y + p3.y) / 3.0;
13        m0 = p1.x * p2.y + p2.x * p3.y + p3.x * p1.y - p1.y * p2.x - p2.y * p3.x
            - p3.y * p1.x;
14        if (cmp(m + m0,0.0) == 0)
15            m0 += eps;
16        p.x = (m * p.x + m0 * p0.x) / (m + m0);
17        p.y = (m * p.y + m0 * p0.y) / (m + m0);
18        m = m + m0;
19        p2 = p3;
20    }
21    return p;
22 }

```

### 7.3 KD树

查找某个点距离最近的点，基本思想是每次分治把点分成两部分，建议按照坐标规模决定是垂直划分还是水平划分，查找时先往分到的那一部分查找，然后根据当前最优答案决定是否去另一个区间查找。

```

1 bool Div[MaxN];
2 void BuildKD(int deep,int l, int r, Point p[])\记得备份一下P
3 {
4     if (l > r) return;
5     int mid = l + r >> 1;
6     int minX, minY, maxX, maxY;
7     minX = min_element(p + l, p + r + 1, cmpX)->x;
8     minY = min_element(p + l, p + r + 1, cmpY)->y;
9     maxX = max_element(p + l, p + r + 1, cmpX)->x;
10    maxY = max_element(p + l, p + r + 1, cmpY)->y;
11    Div[mid] = (maxX - minX >= maxY - minY);
12    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
13    BuildKD(l, mid - 1, p);
14    BuildKD(mid + 1, r, p);
15 }
16
17 long long res;
18 void Find(int l, int r, Point a, Point p[])\查找
19 {
20     if (l > r) return;
21     int mid = l + r >> 1;
22     long long dist = dist2(a, p[mid]);
23     if (dist > 0)//如果有重点不能这样判断
24         res = min(res, dist);
25     long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
26     int l1, l2, r1, r2;
27     l1 = l, l2 = mid + 1;
28     r1 = mid - 1, r2 = r;
29     if (d > 0)
30         swap(l1, l2), swap(r1, r2);
31     Find(l1, r1, a, p);
32     if (d * d < res)
33         Find(l2, r2, a, p);
34 }

```

### 7.3.1 例题

查询一个点为中心的给定正方形内所有点并删除（2012金华网赛A）

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <cmath>
6  #include <queue>
7  using namespace std;
8
9  const int MaxN = 100000;
10 struct Point
11 {
12     int x,y,r;
13     int id;
14     bool del;
15 };
16
17 int cmpTyp;
18 bool cmp(const Point& a,const Point& b)
19 {
20     if (cmpTyp == 0)
21         return a.x < b.x;
22     else
23         return a.y < b.y;
24 }
25
26 int cnt[MaxN];
27 bool Div[MaxN];
28 int minX[MaxN],minY[MaxN],maxX[MaxN],maxY[MaxN];
29 void BuildKD(int l,int r,Point p[])
30 {
31     if (l > r) return;
32     int mid = l+r>>1;
33     cmpTyp = 0;
34     minX[mid] = min_element(p+l,p+r+1,cmp)->x;
35     maxX[mid] = max_element(p+l,p+r+1,cmp)->x;
36     cmpTyp = 1;
37     minY[mid] = min_element(p+l,p+r+1,cmp)->y;
38     maxY[mid] = max_element(p+l,p+r+1,cmp)->y;
39
40     cnt[mid] = r-l+1;
41     cmpTyp = Div[mid] = (maxX[mid]-minX[mid] < maxY[mid]-minY[mid]);
42     nth_element(p+l,p+mid,p+r+1,cmp);
43     BuildKD(l,mid-1,p);
44     BuildKD(mid+1,r,p);
45 }
46
47 queue<int> Q;
48 int Find(int l,int r,Point a,Point p[])
49 {
50     if (l > r) return 0;
51     int mid = l+r>>1;
52     if (cnt[mid] == 0) return 0;
53
54     if (maxX[mid] < a.x-a.r ||
55         minX[mid] > a.x+a.r ||
56         maxY[mid] < a.y-a.r ||
57         minY[mid] > a.y+a.r)
58         return 0;
59 }
```

```

60     int totdel = 0;
61
62     if (p[mid].del == false)
63         if (abs(p[mid].x-a.x) <= a.r && abs(p[mid].y-a.y) <= a.r)
64             {
65                 p[mid].del = true;
66                 Q.push(p[mid].id);
67                 totdel++;
68             }
69
70     totdel += Find(l,mid-1,a,p);
71     totdel += Find(mid+1,r,a,p);
72
73     cnt[mid] -= totdel;
74
75     return totdel;
76 }
77
78 Point p[MaxN],tp[MaxN];
79 int n;
80
81 int main()
82 {
83     int cas = 1;
84     while (true)
85     {
86         scanf("%d",&n);
87         if (n == 0) break;
88
89         for (int i = 0;i < n;i++)
90         {
91             p[i].id = i;
92             int tx,ty;
93             scanf("%d%d%d",&tx,&ty,&p[i].r);
94             p[i].x = tx-ty;
95             p[i].y = tx+ty;
96             p[i].del = false;
97             tp[i] = p[i];
98         }
99         BuildKD(0,n-1,tp);
100
101         printf("Case_#%d:\n",cas++);
102         int q;
103         scanf("%d",&q);
104         for (int i = 0;i < q;i++)
105         {
106             int id;
107             scanf("%d",&id);
108             int res = 0;
109             id--;
110             Q.push(id);
111             while (!Q.empty())
112             {
113                 int now = Q.front();
114                 Q.pop();
115                 if (p[now].del == true) continue;
116                 p[now].del = true;
117                 res += Find(0,n-1,p[now],tp);
118             }
119             printf("%d\n",res);
120         }
121     }
122     return 0;

```

## 7.4 半平面交

直线左边代表有效区域。

```

1 bool HPIcmp(Line a, Line b)
2 {
3     if (fabs(a.k - b.k) > eps) return a.k < b.k;
4     return ((a.s - b.s) * (b.e-b.s)) < 0;
5 }
6
7 Line Q[100];
8 void HPI(Line line[], int n, Point res[], int &resn)
9 {
10     int tot = n;
11     sort(line, line + n, HPIcmp);
12     tot = 1;
13     for (int i = 1; i < n; i++)
14         if (fabs(line[i].k - line[i - 1].k) > eps)
15             line[tot++] = line[i];
16     int head = 0, tail = 1;
17     Q[0] = line[0];
18     Q[1] = line[1];
19     resn = 0;
20     for (int i = 2; i < tot; i++)
21     {
22         if (fabs((Q[tail].e-Q[tail].s) * (Q[tail - 1].e-Q[tail - 1].s)) <
23             eps ||
24             fabs((Q[head].e-Q[head].s) * (Q[head + 1].e-Q[head + 1].s))
25                 < eps)
26             return;
27         while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s) * (line[i]
28             ].e-line[i].s)) > eps)
29             tail--;
30         while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s) * (line[i]
31             ].e-line[i].s)) > eps)
32             head++;
33         Q[++tail] = line[i];
34     }
35     while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) * (Q[head].e-
36         Q[head].s)) > eps)
37         tail--;
38     while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q[tail].e-
39         Q[tail].s)) > eps)
40         head++;
41     if (tail <= head + 1) return;
42     for (int i = head; i < tail; i++)
43         res[resn++] = Q[i] & Q[i + 1];
44     if (head < tail + 1)
45         res[resn++] = Q[head] & Q[tail];
46 }
```

## 7.5 凸包

得到的凸包按照逆时针方向排序。

```

1 bool GScmp(Point a, Point b)
2 {
3     if (fabs(a.x - b.x) < eps)
4         return a.y < b.y - eps;
5     return a.x < b.x - eps;
6 }
7
```



```

8 void GS(Point p[], int n, Point res[], int &resn)
9 {
10     resn = 0;
11     int top = 0;
12     sort(p, p + n, GScmp);
13     for (int i = 0; i < n;)
14         if (resn < 2 || (res[resn - 1] - res[resn - 2]) * (p[i] - res[resn - 1]) > eps)
15             res[resn++] = p[i++];
16         else
17             --resn;
18     top = resn - 1;
19     for (int i = n - 2; i >= 0;)
20         if (resn < top + 2 || (res[resn - 1] - res[resn - 2]) * (p[i] - res[resn - 1]) > eps)
21             res[resn++] = p[i--];
22         else
23             --resn;
24     resn--;
25     if (resn < 3)    resn = 0;
26 }

```

## 7.6 直线与凸包求交点

复杂度 $O(\log n)$ 。

需要先预处理几个东西。

```

1 //二分[la,lb]这段区间那条边与line相交
2 int Gao(int la,int lb,Line line)
3 {
4     if (la > lb)
5         lb += n;
6     int l = la,r = lb,mid;
7     while (l < r)
8     {
9         mid = l+r+1>>1;
10        if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(p[mid]-line.s),0) >= 0)
11            l = mid;
12        else
13            r = mid-1;
14    }
15    return l%n;
16 }
17 //求l与凸包的交点
18
19 //先调用Gettheta预处理出凸包每条边的斜率，然后处理成升序排列
20 double theta[maxn];
21
22 void Gettheta()
23 {
24     for (int i = 0;i < n;i++)
25     {
26         Point v = p[(i+1)%n]-p[i];
27         theta[i] = atan2(v.y,v.x);
28     }
29     for (int i = 1;i < n;i++)
30         if (theta[i-1] > theta[i]+eps)
31             theta[i] += 2*pi;
32 }
33
34 double Calc(Line l)

```

```

35 {
36     double tnow;
37     Point v = l.e-l.s;
38     tnow = atan2(v.y,v.x);
39     if (cmp(tnow,theta[0]) < 0)        tnow += 2*pi;
40     int pl = lower_bound(theta,theta+n,tnow)-theta;
41     tnow = atan2(-v.y,-v.x);
42     if (cmp(tnow,theta[0]) < 0)        tnow += 2*pi;
43     int pr = lower_bound(theta,theta+n,tnow)-theta;
44     //pl和pr是在l方向上距离最远的点对
45     pl = pl%n;
46     pr = pr%n;
47
48     if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
49         return 0.0;
50
51     int xa = Gao(pl,pr,l);
52     int xb = Gao(pr,pl,l);
53
54     if (xa > xb)        swap(xa,xb);
55     //与[xa,xa+1]和[xb,xb+1]这两条线段相交
56
57     if (cmp(v*(p[xa+1]-p[xa]),0) == 0)    return 0.0;
58     if (cmp(v*(p[xb+1]-p[xb]),0) == 0)    return 0.0;
59
60     Point pa,pb;
61     pa = Line(p[xa],p[xa+1])&l;
62     pb = Line(p[xb],p[xb+1])&l;
63     //题目: 求直线切凸包得到的两部分的面积
64     double area0 = sum[xb]-sum[xa+1]+(pa*p[xa+1])/2.0+(p[xb]*pb)/2.0+(pb*pa)
65         /2.0;
66     double area1 = sum[xa+n]-sum[xb+1]+(pb*p[xb+1])/2.0+(p[xa]*pa)/2.0+(pa*
67         pb)/2.0;
68     return min(area0,area1);
69 }

```

## 7.7 三维凸包

暴力写法

```

1  #define eps 1e-7
2  #define MAXV 505
3
4  struct pt
5  {
6      double x, y, z;
7      pt() {}
8      pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
9      pt operator - (const pt p1)
10     {
11         return pt(x - p1.x, y - p1.y, z - p1.z);
12     }
13     pt operator * (pt p)
14     {
15         return pt(y*p.z-z*p.y, z*p.x-x*p.z, x*p.y-y*p.x);
16     }
17     double operator ^ (pt p)
18     {
19         return x*p.x+y*p.y+z*p.z;
20     }
21 };
22 struct _3DCH

```

```

23 {
24     struct fac
25     {
26         int a, b, c;
27         bool ok;
28     };
29     int n;
30     pt P[MAXV];
31     int cnt;
32     fac F[MAXV*8];
33     int to[MAXV][MAXV];
34     double vlen(pt a)
35     {
36         return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
37     }
38     double area(pt a, pt b, pt c)
39     {
40         return vlen((b-a)*(c-a));
41     }
42     double volume(pt a, pt b, pt c, pt d)
43     {
44         return (b-a)*(c-a)^(d-a);
45     }
46     double ptof(pt &p, fac &f)
47     {
48         pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
49         return (m * n) ^ t;
50     }
51     void deal(int p, int a, int b)
52     {
53         int f = to[a][b];
54         fac add;
55         if (F[f].ok)
56         {
57             if (ptof(P[p], F[f]) > eps)
58                 dfs(p, f);
59             else
60             {
61                 add.a = b, add.b = a, add.c = p, add.ok = 1;
62                 to[p][b] = to[a][p] = to[b][a] = cnt;
63                 F[cnt++] = add;
64             }
65         }
66     }
67     void dfs(int p, int cur)
68     {
69         F[cur].ok = 0;
70         deal(p, F[cur].b, F[cur].a);
71         deal(p, F[cur].c, F[cur].b);
72         deal(p, F[cur].a, F[cur].c);
73     }
74     bool same(int s, int t)
75     {
76         pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
77         return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a, b, c
78             , P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
79     }
80     void construct()
81     {
82         cnt = 0;
83         if (n < 4)
84             return;

```

```

85     bool sb = 1;
86     for (int i = 1; i < n; i++)
87     {
88         if (vlen(P[0] - P[i]) > eps)
89         {
90             swap(P[1], P[i]);
91             sb = 0;
92             break;
93         }
94     }
95     if (sb) return;
96     sb = 1;
97     for (int i = 2; i < n; i++)
98     {
99         if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
100        {
101            swap(P[2], P[i]);
102            sb = 0;
103            break;
104        }
105    }
106    if (sb) return;
107    sb = 1;
108    for (int i = 3; i < n; i++)
109    {
110        if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps)
111        {
112            swap(P[3], P[i]);
113            sb = 0;
114            break;
115        }
116    }
117    if (sb) return;
118    fac add;
119    for (int i = 0; i < 4; i++)
120    {
121        add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok = 1;
122        if (ptof(P[i], add) > 0)
123            swap(add.b, add.c);
124        to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
125        F[cnt++] = add;
126    }
127    for (int i = 4; i < n; i++)
128    {
129        for (int j = 0; j < cnt; j++)
130        {
131            if (F[j].ok && ptof(P[i], F[j]) > eps)
132            {
133                dfs(i, j);
134                break;
135            }
136        }
137    }
138    int tmp = cnt;
139    cnt = 0;
140    for (int i = 0; i < tmp; i++)
141    {
142        if (F[i].ok)
143        {
144            F[cnt++] = F[i];
145        }
146    }
147    }

```

```

148 //表面积
149 double area()
150 {
151     double ret = 0.0;
152     for (int i = 0; i < cnt; i++)
153     {
154         ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
155     }
156     return ret / 2.0;
157 }
158 //体积
159 double volume()
160 {
161     pt O(0, 0, 0);
162     double ret = 0.0;
163     for (int i = 0; i < cnt; i++)
164     {
165         ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
166     }
167     return fabs(ret / 6.0);
168 }
169 //表面三角形数
170 int facetCnt_tri()
171 {
172     return cnt;
173 }
174 //表面多边形数
175 int facetCnt()
176 {
177     int ans = 0;
178     for (int i = 0; i < cnt; i++)
179     {
180         bool nb = 1;
181         for (int j = 0; j < i; j++)
182         {
183             if (same(i, j))
184             {
185                 nb = 0;
186                 break;
187             }
188         }
189         ans += nb;
190     }
191     return ans;
192 }
193
194 pt Fc[MAXV*8];
195 double V[MAXV*8];
196 pt Center()//重心
197 {
198     pt O(0,0,0);
199     for (int i = 0; i < cnt; i++)
200     {
201         Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;
202         Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
203         Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
204         V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
205     }
206     pt res = Fc[0],tmp;
207     double m = V[0];
208     for (int i = 1; i < cnt; i++)
209     {
210         if (fabs(m+V[i]) < eps)

```

```

211         V[i] += eps;
212         tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
213         tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
214         tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
215         m += V[i];
216         res = tmp;
217     }
218     return res;
219 }
220 };
221
222 _3DCH hull;
223
224 int main()
225 {
226     while (scanf("%d",&hull.n) != EOF)
227     {
228         for (int i = 0; i < hull.n; i++)
229             scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i].z);
230         hull.construct();
231     }
232     return 0;
233 }

```

## 7.8 旋转卡壳

“对踵”

### 7.8.1 单个凸包

```

1 void solve(Point p[],int n)
2 {
3     Point v;
4     int cur = 1;
5     for (int i = 0;i < n;i++)
6     {
7         v = p[i]-p[(i+1)%n];
8         while (v*(p[(cur+1)%n]-p[cur]) < 0)
9             cur = (cur+1)%n;
10        //p[cur] -> p[i]
11        //p[cur] -> p[i+1]
12        //p[cur] -> (p[i],p[i+1])
13    }
14 }

```

### 7.8.2 两个凸包

注意初始点的选取，代码只是个示例。

有时候答案需要取solve(p0,n,p1,m)和solve(p1,m,p0,n)的最优值。

```

1 void solve(Point p0[],int n,Point p1[],int m)
2 {
3     Point v;
4     int cur = 0;
5     for (int i = 0;i < n;i++)
6     {
7         v = p0[i]-p0[(i+1)%n];
8         while (v*(p1[(cur+1)%m]-p1[cur]) < 0)
9             cur = (cur+1)%m;
10        //p1[cur] -> p0[i]
11        //p1[cur] -> p0[i+1]
12        //p1[cur] -> (p0[i],p0[i+1])

```

```

13     }
14 }

```

### 7.8.3 外接矩形

```

1 void solve()
2 {
3     resa = resb = 1e100;
4     double dis1,dis2;
5     Point xp[4];
6     Line l[4];
7     int a,b,c,d;
8     int sa,sb,sc,sd;
9     a = b = c = d = 0;
10    sa = sb = sc = sd = 0;
11    Point va,vb,vc,vd;
12    for (a = 0; a < n; a++)
13    {
14        va = Point(p[a],p[(a+1)%n]);
15        vc = Point(-va.x,-va.y);
16        vb = Point(-va.y,va.x);
17        vd = Point(-vb.x,-vb.y);
18        if (sb < sa)
19        {
20            b = a;
21            sb = sa;
22        }
23        while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
24        {
25            b = (b+1)%n;
26            sb++;
27        }
28        if (sc < sb)
29        {
30            c = b;
31            sc = sb;
32        }
33        while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
34        {
35            c = (c+1)%n;
36            sc++;
37        }
38        if (sd < sc)
39        {
40            d = c;
41            sd = sc;
42        }
43        while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
44        {
45            d = (d+1)%n;
46            sd++;
47        }
48
49        //卡在p[a],p[b],p[c],p[d上]
50        sa++;
51    }
52 }

```

## 8 搜索

### 8.1 Dancing Links

仰慕罗神。

```
1 void remove1(int col)
2 {
3     int i,j;
4     L[R[col]]=L[col];
5     R[L[col]]=R[col];
6     for(i=D[col];i!=col;i=D[i])
7     {
8         L[R[i]]=L[i];
9         R[L[i]]=R[i];
10    }
11 }
12 void remove2(int col)
13 {
14     int i,j;
15     L[R[col]]=L[col];
16     R[L[col]]=R[col];
17     for(i=D[col];i!=col;i=D[i])
18     {
19         for(j=R[i];j!=i;j=R[j])
20         {
21             U[D[j]]=U[j];
22             D[U[j]]=D[j];
23             --nk[C[j]];
24         }
25     }
26 }
27 void resume1(int col)
28 {
29     int i,j;
30     for(i=U[col];i!=col;i=U[i])
31     {
32         L[R[i]]=i;
33         R[L[i]]=i;
34     }
35     L[R[col]]=col;
36     R[L[col]]=col;
37 }
38 void resume2(int col)
39 {
40     int i,j;
41     for(i=U[col];i!=col;i=U[i])
42     {
43         for(j=L[i];j!=i;j=L[j])
44         {
45             ++nk[C[j]];
46             U[D[j]]=j;
47             D[U[j]]=j;
48         }
49     }
50     L[R[col]]=col;
51     R[L[col]]=col;
52 }
53 int h()
54 {
55     bool vis[100];
56     memset(vis,false,sizeof(vis));
57     int i,j,k,res=0,mi,col;
```



```

58     while(1)
59     {
60         mi=inf;
61         for(i=R[head];i!=head&&i<=2*n;i=R[i])
62             if(mi>nk[i]&&!vis[i])
63             {
64                 mi=nk[i];
65                 col=i;
66             }
67         if(mi==inf)
68             break;
69         res++;vis[col]=true;
70         for(j=D[col];j!=col;j=D[j])
71             for(k=R[j];k!=j;k=R[k])
72             {
73                 if(C[k]>2*n)
74                     continue;
75                 vis[C[k]]=true;
76             }
77     }
78     return res;
79 }
80 bool DLX(int d,int deep)
81 {
82     if(d+h(>deep) return false;
83     if(R[head]==head||R[head]>2*n)
84         return true;
85     if(d>=deep)
86         return false;
87     int col,ma=inf;
88     int i,j;
89     for(i=R[head];i!=head&&i<=2*n;i=R[i])
90         if(nk[i]<ma)
91         {
92             col=i;
93             ma=nk[i];
94         }
95     remove1(col);
96     for(i=D[col];i!=col;i=D[i])
97     {
98         int flag=1;
99         for(j=R[i];j!=R[j])
100         {
101             if(j==R[i]&&!flag)
102                 break;
103             U[D[j]]=U[j];
104             D[U[j]]=D[j];
105             if(C[j]>2*n)
106                 remove2(C[j]);
107             else
108                 remove1(C[j]);
109             flag=0;
110         }
111         if(DLX(d+1,deep))
112             return true;
113         flag=1;
114         for(j=L[i];j!=L[j])
115         {
116             if(j==L[i]&&!flag)
117                 break;
118             if(C[j]>2*n)
119                 resume2(C[j]);
120             else

```

```

121         resume1(C[j]);
122         U[D[j]]=j;
123         D[U[j]]=j;
124         flag=0;
125     }
126 }
127 resume1(col);
128 return false;
129 }

```

## 9 杂物

### 9.1 高精度数

支持乘以整数和加法。

```
1 struct BigInt
2 {
3     const static int mod = 100000000;
4     int a[600], len;
5     BigInt () {}
6     BigInt (int v)
7     {
8         len = 0;
9         do
10         {
11             a[len++] = v%mod;
12             v /= mod;
13         }while(v);
14     }
15     BigInt operator *(const int& b) const
16     {
17         BigInt res;
18         res.len = len;
19         for (int i = 0; i <= len; ++i)
20             res.a[i] = 0;
21         for (int i = 0; i < len; ++i)
22         {
23             res.a[i] += a[i]*b;
24             res.a[i+1] += res.a[i]/mod;
25             res.a[i] %= mod;
26         }
27         if (res.a[len] > 0) res.len++;
28         return res;
29     }
30     BigInt operator +(const BigInt& b) const
31     {
32         BigInt res;
33         res.len = max(len, b.len);
34         for (int i = 0; i <= res.len; ++i)
35             res.a[i] = 0;
36         for (int i = 0; i < res.len; ++i)
37         {
38             res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0);
39             res.a[i+1] += res.a[i]/mod;
40             res.a[i] %= mod;
41         }
42         if (res.a[res.len] > 0) res.len++;
43         return res;
44     }
45     void output()
46     {
47         printf("%d", a[len-1]);
48         for (int i = len-2; i >= 0; --i)
49             printf("%08d", a[i]);
50         printf("\n");
51     }
52 };
```

### 9.2 整数外挂

```
1 int wg;
```

```

2 char ch;
3 bool ng;
4
5 inline int readint()
6 {
7     ch = getchar();
8     while (ch != '-' && (ch < '0' || ch > '9')) ch = getchar();
9     if (ch == '-')
10    {
11        ng = true;
12        ch = getchar();
13    }
14    else
15        ng = false;
16    wg = ch-'0';
17    ch = getchar();
18    while (ch >= '0' && ch <= '9')
19    {
20        wg = wg*10+ch-'0';
21        ch = getchar();
22    }
23    if (ng == true) wg = -wg;
24    return wg;
25 }

```

## 9.3 Java

### 9.3.1 优先队列

```

1 PriorityQueue queue = new PriorityQueue( 1, new Comparator()
2 {
3     public int compare( Point a, Point b )
4     {
5         if( a.x < b.x || a.x == b.x && a.y < b.y )
6             return -1;
7         else if( a.x == b.x && a.y == b.y )
8             return 0;
9         else
10            return 1;
11    }
12 });

```

### 9.3.2 Map

```

1 Map map = new HashMap();
2 map.put("sa","dd");
3 String str = map.get("sa").toString;
4
5 for(Object obj : map.keySet()){
6     Object value = map.get(obj );
7 }

```

### 9.3.3 sort

```

1 static class cmp implements Comparator
2 {
3     public int compare(Object o1, Object o2)
4     {
5         BigInteger b1=(BigInteger)o1;
6         BigInteger b2=(BigInteger)o2;
7         return b1.compareTo(b2);
8     }
9 }
10 public static void main(String[] args) throws IOException
11 {

```

```
12 | Scanner cin = new Scanner(System.in);
13 | int n;
14 | n=cin.nextInt();
15 | BigInteger[] seg = new BigInteger[n];
16 | for (int i=0;i<n;i++)
17 |     seg[i]=cin.nextBigInteger();
18 | Arrays.sort(seg,new cmp());
19 | }
```